RSA Blind Signatures with Public Metadata

Ghous Amjad Kevin Yeo Moti Yung

Abstract

Anonymous tokens are digital signature schemes that enable an issuer to provider users with signatures without learning the input message or the resulting signature received by the user. These primitives allow applications to propagate trust while simultaneously protecting the identity of the user. Anonymous tokens have become a core component for improving the privacy of several real-world applications including ad measurements, authorization protocols, spam detection and VPNs.

In certain applications, it is natural to associate signatures with specific public metadata ensuring that signatures only propagate trust with respect to only a certain set of scenarios. To solve this, we study the notion of anonymous tokens with public metadata in this work. We present a variant of RSA blind signatures with public metadata such that issuers may only generate signatures that verify for a certain choice of public metadata. We prove the security of our protocol under one-more RSA assumptions with multiple exponents that we introduce. Furthermore, we provide evidence that the concrete security bounds should be nearly identical to standard RSA blind signatures. The protocols in this paper have been proposed as a technical specification in an IRTF internet draft.[1]

1 Introduction

Anonymous tokens are a powerful primitive that have been studied for decades under various names including anonymous credentials and blind signatures. Throughout our work, we will use anonymous tokens and blind signatures interchangeably. They were first introduced by Chaum [15, 16] in the context electronic cash that would be untraceable. At a high level, the idea was that any bank or treasury would be able to attest the validity of money. By using anonymous tokens, the guarantee is that no one would ever be able to trace the usage of the money even if the bank knows the identity of the user that was originally granted the money. This is just one example that exhibits the usefulness of anonymous tokens. In recent years, anonymous tokens have become a core component of many real-world applications including private click measurement [5], privacy-preserving telemetry [23], fraud detection [3], avoiding repeated CAPTCHA solving [19] and web browsing [4, 6] to list some examples.

There are typically three different roles for participants in anonymous token schemes. The user (also commonly referred to as the signature recipient or receiver) is the party that is requesting and receiving the signature. The signer (or issuer) is the party that receives a blind signing request and is responsible for issuing a response that enables the user to recover the final signature. Finally, the last party is the verifier whose responsibility is to check whether a signature is one that was correctly signed by the signer. Anonymous token schemes may be split into two types: designated or public verifiability. In the designated setting, verification requires the usage of the private key and certain parties must be explicitly assigned the role of verifier by being given the private key. In the public verifiability case, anyone can perform verification given they have access to the public key. In our work, we will exclusively focus on public verification. We note that a party may take on more than one role in various applications. For example, a party may be responsible for both issuing signatures as well as verifying their validity.

Anonymous tokens are required to satisfy three important properties: correctness, unforgeability and unlinkability. For correctness, it must be the case that a signature will be successfully accepted by a verifier as long as all parties involved were acting honestly (that is, following the algorithm correctly). Unforgeability

[1]For the accompanying CFRG specification in the IRTF, we point readers to [10].

Google, {gamjad,kwlyeo,moti}@google.com.

describes the property that an adversary should not be able to create valid signatures without interacting with an signer. For example, an adversarial user that performs ℓ blind signing protocols with an signer should not be able to create $\ell+1$ distinct pairs of message and valid signatures of the corresponding message. Both correctness and unforgeability are properties required from any digital signature scheme. The last property of unlinkability formally describes the notion of anonymity with respect to an untrusted signer. In particular, the signer should be unable to link any blind signing request with any final signature. Suppose that the signer answers ℓ blind signing requests. Afterwards, the signer receives the ℓ resulting signatures that are randomly permuted. Then, the signer is unable to link any request with a specific signature with probability better than randomly guessing. In other words, a signature remains anonymous in the view of the adversarial signer that may have maliciously chosen the system's parameters.

In this work, we study a variant of anonymous tokens with public metadata (or partially blind signatures) where each signature will be tied to a specific piece of public metadata. This allows the user and signer to jointly agree on explicitly encoding necessary information into the final signatures. Additionally, the verifier will need to specify the public metadata for which the signature is being verified against. These primitives guarantee that a signature will only be accepted if the verification algorithm is using the public metadata that was originally used to create the signature. The ability to augment signatures with public metadata serves as a way to only propagate trust for a specific subset of settings. As an example, we can consider a product with localization that requires purchasing for each different country and needs to create signatures tied to the purchase country of the user. Anonymous tokens with public metadata would allow tying each signature to the purchase country guaranteeing that signatures are only valid in the country of purchase.

IRTF Draft. The protocols in this paper have already been proposed in an CFRG specification in the IRTF by Amjad, Hendrickson, Wood and Yeo [10] due to its potential for real-world impact. By enabling practical anonymous tokens with public metadata, this work unblocks a significant number of new applications. For example, there are already work streams to build real-world architectures relying on anonymous tokens with public metadata in the IETF Privacy Pass working group (such as [22]). One can view this work as the accompanying academic paper proving formal cryptographic guarantees for the usage of these protocols in real-world applications.

1.1 Technical Overview

Before we present our protocol for RSA blind signatures with public metadata, we first show two failed approaches that provide insights into the design choices of our final algorithm. In each of these approaches, we take an underlying anonymous token scheme and aim to enable public metadata.

One Key for Each Public Metadata. The first idea is to provide isolation between the different choices of public metadata using different keys. Formally, for each potential option of public metadata D, we can generate a new RSA key pair $(\mathsf{pk}_D, \mathsf{sk}_D)$. This is equivalent to having a separate anonymous token system for each choice of public metadata. As the public metadata D is known to the signer and users, each party can use the appropriate keys corresponding to D to performing blind signing and verification.

While the above approach would work, there are several drawbacks to this approach. First, the number of keys that needs to be stored by all parties must be on the order on the number of different choices of public metadata. This is not realistic when the universe of public metadata is very large. Even if the public metadata universe is reasonably sized, the task of key management becomes less than ideal due to the existence of multiple keys at once. The other disadvantage of this approach is that the universe of possible public metadata must be fixed ahead of time as a new key pair must be generated for each choice of public metadata. If the universe of public metadata changes, all parties in the system must also update their key sets to include the new public and private keys associated with the new public metadata.

Given the above, we want an anonymous token with public metadata scheme that does not require multiple keys while still providing the flexibility of a potentially changing set of public metadata.

Combining Public Metadata and Message. Another approach may be to try and use the message to encode the public metadata. For example, the public metadata D can be appended to the message M that needs to be signed to obtain the new message $M' = M \mid\mid D$. Afterwards, the anonymous token scheme remains identical with using the new message M'.

At first, this seems to achieve the desired goal as the privacy of the signature and message follows directly from the underlying anonymous token scheme. Furthermore, if one attempts to use the wrong public metadata, verification will clearly fail as the input message is incorrect. Unfortunately, this approach would only work in the case that we are considering signature schemes that do not provide anonymity. In particular, the metadata must be publicly known to the signer during the signature process. For the setting without anonymity, the user can simply upload the augmented message $M' = M \mid\mid D$ to the signer where the signer can check that the public metadata D before signing the message M'. As the signer views the message in plaintext, it will be impossible to obtain anonymity.

If we wish to maintain anonymity, then the message $M' = M \mid\mid D$ cannot be sent to the signer in plaintext. Therefore, the signer can no longer verify that they are signing with respect to a specific piece of public metadata D. As an example, consider an application that requires associating each signature to a specific area using public metadata. Even if the signer wishes to only create signatures for a specific public metadata D (such as a specific country), the signer is unable to check the public metadata for the specific blind signing request. Therefore, this approach fails to satisfy the necessary unforgeability requirements for adversarial users.

Our Protocol. Using the above two failed approaches, we can determine some requirements necessary for our construction. First, we should only use a single key independent of the public metadata universe. Additionally, we still need the public metadata to be viewable by the signer during blind signing. To achieve both of these goals, we will build upon standard RSA blind signatures.

We first revisit the original RSA blind signature protocol. Recall that standard RSA blind signatures utilize a modulus that is the product of two primes $N = pq$ along with a public exponent e and a private exponent d such that $d = e^{-1} \mod \phi(N)$.[2] The signer's private key consists of d while the public key is (N, e). In the rest of the description, all operations are done in $\mathbb{Z}_N^*$. To perform blind signing for a message M, the user computes $A = H_{\mathcal{M}}(M) \cdot R^e$ where $H_{\mathcal{M}}(M)$ hashes the message M to an element of $\mathbb{Z}_N^*$ and R is a uniformly random element of $\mathbb{Z}_N^*$. This is sent to the signer who signs by computing $B = A^d = H_{\mathcal{M}}(M)^d \cdot R^{ed} = H_{\mathcal{M}}(M)^d \cdot R$. Finally, the user computes $B \cdot R^{-1} = H_{\mathcal{M}}(M)^d \cdot R \cdot R^{-1} = H_{\mathcal{M}}(M)^d$ that is the final signature. To verify any signature S for message M using public key (e, N), one simply computes S^d and checks if it equals to $H_{\mathcal{M}}(M)$.[3]

We take insight from the first failed approach that used different keys for each metadata. Rather than using different keys, we can try to use a hash function to enable generating keys for each metadata. For RSA blind signatures, we will use a hash function H_{MD} to generate public and private exponents that are specific to some public metadata D. In more detail, we will set $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$ as the public exponent. Blind signing for the user remains similar using e_{MD} as opposed to e to obtain $A = H_{\mathcal{M}}(M \mid\mid D) \cdot R^{e_{\mathsf{MD}}}$ with the exception that we also put the public metadata D into the hash now. Now, the signer receives both the blind signing request A as well as the public metadata D. First, the signer computes $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. We augment the private key to contain $\phi(N)$ allowing the signer to compute the inverse $d_{\mathsf{MD}} = (e_{\mathsf{MD}})^{-1} \mod \phi(N)$. Afterwards, the signer uses d_{MD} as the private exponent to return $B = H_{\mathcal{M}}(M \mid\mid D)^{d_{\mathsf{MD}}} \cdot R$ to the user. Finally, the user removes R to obtain $H_{\mathcal{M}}(M \mid\mid D)^{d_{\mathsf{MD}}}$ that is the final signature.

Unfortunately, the above construction has a slight problem. In particular, we assumed that the output $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$ is always invertible modulo $\phi(N)$ and, thus, co-prime to $\phi(N)$. For standard RSA modulus $N = pq$ where p and q are distinct primes, it is not necessarily the case that a random element would be invertible modulo $\phi(N) = (p-1) \cdot (q-1)$. To solve this problem, we can use strong RSA moduli where we require that $N = pq$ such that both p and q are safe primes meaning that $p = 2p' + 1$ and $q = 2q' + 1$ such that both p' and q' are also prime numbers. If N is a strong RSA modulus, then we know that $\phi(N) = (p-1) \cdot (q-1) = 4p'q'$. Therefore, an element $x \in \mathbb{Z}_{\phi(N)}$ has an inverse as long as x is odd and not divisible by p' and q'. If we assume that both p and q are λ-bit prime numbers, then we can guarantee

[2]Euler's totient function $\phi(N)$ counts the positive integers up to the given integer N that are relatively prime to N.

[3]For sake of simple exposition, we assumed that $H_{\mathcal{M}}$ is deterministic. However, there exists other message encodings that are randomized with more complex verification algorithms. See Section 5.1 for more details.

that the output of H_{MD} is always invertible in $\mathbb{Z}_{\phi(N)}$ using the following modifications. First, we ensure that H_{MD} always outputs an odd number. Next, we make sure that H_{MD} always outputs elements of length at most $\lambda - 3$ bits. As p and q are λ bits, we know that both p' and q' are at least $\lambda - 2$ bits. As a result, we can guarantee that H_{MD} is always invertible modulo $\phi(N)$ as it is odd and always smaller than both p' and q'. We note that similar ideas were previously presented in [9, 8]. However, to our knowledge, no formal security proofs have been provided for these constructions.

One-More RSA Inversion with Multiple Exponents. Before proving security of our protocol, we first need to define an extension of the RSA assumption using the "one-more" style techniques of Bellare, Namprempre, Pointcheval and Semanko [12]. We explore several natural ways to define RSA assumptions with respect to multiple exponents before arriving at the weakest form that may be used to prove security of our protocol. To our knowledge, this is the first exploration of one-more RSA inversion assumptions with multiple exponents. Finally, we also explore connections with various natural definitions of one-more strong[4] RSA type assumptions.

Security of Our Protocol. Finally, we prove the security of our new protocol with public metadata. We start by considering a non-blind variant that may be used as a standard signature scheme. We are able to prove the concrete security of our new protocol from the RSA assumption. In particular, we show that our new protocol has almost identical concrete security guarantees for unforgeability as standard RSA signatures. The only security loss is an additive factor of approximately $2^{O(-\lambda)}$ depending on the output length λ of an underlying hash function. We note that our subtle modification where the underlying hash $H_{\mathcal{M}}(M \mid\mid D)$ includes both the message M and the public metadata D is integral in our security proof.

Next, we move onto proving the security of our main protocol of RSA blind signatures with public metadata protocol. Using the one-more multi-exponent RSA inversion assumptions, we are able to prove the unforgeability of our protocol. For anonymity, we adapt the techniques introduced by Lysyanskaya [25] to ensure and prove that our protocols satisfy unlinkability even in the presence of keys that are generated by a malicious party.

1.2 Related Work

BLS with Public Metadata. We note that a prior work of Silde and Strand [32] also studied the notion of anonymous tokens with public metadata where several constructions were presented. For the setting of public verifiability, the authors presented a construction from pairings based on the Boneh, Lynn and Shacham (BLS) signature scheme [14] along with ideas from other works including [33, 19]. There are several differences between our construction and this one. First, our construction uses RSA-based cryptography while the BLS-based scheme uses pairing-based cryptography. In general, RSA-based cryptography is more readily available on all platforms compared to pairing-based cryptography (for example, this is a core reason why VPN by GoogleOne [6] chose RSA blind signatures). In terms of efficiency, our RSA blind signature with public metadata scheme will incur smaller computation, but larger signatures and more communication during blind signing compared to the BLS-based scheme. For cases when computation may be limited (such as devices with limited computational power), our schemes may be the better option.

RSA Blind Signatures with Strong Moduli. We note that similar variants of RSA blind signatures has appeared in the past by Abe and Fujisaki [9] as well as Abe and Camenisch [8]. However, the security was proven heuristically and predated the standard definitions used in anonymous tokens. In our work, we formally prove the security of our protocols along with concrete security bounds. We also note that other works also used strong RSA modulus such as threshold signatures [30] and integer commitments [18].

Anonymous Tokens with Private Metadata. Recent work [24] has also studied the case where anonymous tokens are augmented with a single private metadata bit. In this case, the metadata bit is explicitly specified only by the signer during the blind signing process. Furthermore, the verifier is also able to view the metadata bit. In particular, the goal is to hide the metadata bit from the user.

[4]In the strong RSA assumption, the adversary gets to pick its own public exponent (Section 4.3). This is unrelated to a strong RSA modulus that is the product of two safe primes.

2 Anonymous Tokens

We present the formal definitions for anonymous token schemes. Our definitions will work for both anonymous and non-anonymous protocols with small modifications. We will solely focus on the settings of public metadata and public verifiability (that is, verification only requires the public key). We chose this definition to be compatible with RSA signatures as well as the blind version. Other variants of anonymous token schemes exist such as those only allowing verification using the private key (such as [19, 32]) as well as enabling private metadata (see [24]).

Definition 1. *A token scheme with public metadata* Tok *that is publicly verifiable is a tuple of efficient algorithms* $\mathsf{Tok} = (\mathsf{Setup}, \mathsf{Blind}, \mathsf{Sign}, \mathsf{Finalize}, \mathsf{Verify})$.

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Tok.Setup}(1^\lambda)$*: The setup algorithm receives the security parameter* λ *as input and outputs a pair of public and private keys* $(\mathsf{pk}, \mathsf{sk})$.
2. $(\mathsf{st}, B_M) \leftarrow \mathsf{Tok.Blind}(M, D, \mathsf{pk})$*: The blinding protocol is run by the user who receives the plaintext message* M*, public metadata* D *and public key* pk*. The output* B_M *is a blinded version of the message* M *under public metadata* D *and some state* st.
3. $S' \leftarrow \mathsf{Tok.Sign}(B_M, D, \mathsf{pk}, \mathsf{sk})$*: The signing protocol is run by the signer who receives the blinded message* B_M*, public metadata* D *and both public and private keys* $(\mathsf{pk}, \mathsf{sk})$*. The output* S' *is a signature on the blinded message* B_M *under public metadata* D *which needs to be finalized by the user.*
4. $S \leftarrow \mathsf{Tok.Finalize}(\mathsf{st}, S', M, D, \mathsf{pk})$*: The user runs the* $\mathsf{Finalize}$ *protocol on the user state* st*, the signer's response* S'*, plaintext message* M*, public metadata* D *and public key* pk*. The output* S *is a signature of the message* M *under public metadata* D.
5. $b \leftarrow \mathsf{Tok.Verify}(S, M, D, \mathsf{pk})$*: The verification algorithm receives the signature* S*, the plaintext message* M*, public metadata* D *and public key* pk *and outputs a bit* $b \in \{0, 1\}$.

The token scheme Tok *satisfies the correctness properties if, for all choices of messages* M *and public metadata* D*, the following holds:*

$$\Pr\left[\mathsf{Tok.Verify}(S, M, D, \mathsf{pk}) = 1 \;:\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Tok.Setup}(1^\lambda) \\ (\mathsf{st}, B_M) \leftarrow \mathsf{Tok.Blind}(M, D, \mathsf{pk}) \\ S' \leftarrow \mathsf{Tok.Sign}(B_M, D, \mathsf{pk}, \mathsf{sk}) \\ S \leftarrow \mathsf{Tok.Finalize}(\mathsf{st}, S', M, D, \mathsf{pk}) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda).$$

We note that the $\mathsf{Blind}, \mathsf{Sign}, \mathsf{Finalize}$ protocols can be viewed as a single round-trip protocol between the user and signer. One could generalize the above definition to encompass multiple round, interactive protocols. However, the structure of the $\mathsf{Blind}, \mathsf{Sign}, \mathsf{Finalize}$ protocol will be useful for proving unforgeability of our schemes. Therefore, we only focus on this structure throughout our work.

Definition for Non-Anonymous Tokens. To obtain the standard definition of non-anonymous tokens, we note that we can simply restrict the functionality of Blind and $\mathsf{Finalize}$ to perform no operations. We will use this when evaluating the concrete security of RSA signatures with public metadata.

1. $\mathsf{Tok.Blind}(M, D, \mathsf{pk})$:
   (a) Return $(\mathsf{st} \leftarrow \perp, B_M \leftarrow M)$.
2. $\mathsf{Tok.Finalize}(\mathsf{st}, S', M, D, \mathsf{pk})$:
   (a) Return $S \leftarrow S'$.

```
Game $\mathsf{G}^{\mathsf{SOMUF}}_{\mathsf{Tok},\mathcal{A}}(\lambda)$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Tok.Setup}(1^\lambda)$
  Initialize $\mathsf{cnt}_D \leftarrow 0$ for all choices of public metadata $D$.
  $D, (S_i, M_i)_{i \in [x]} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{pk})$
  Return 1 if and only if all the following hold:
    - $\mathsf{cnt}_D < x$
    - $\forall i \neq j \in [x], S_i \neq S_j \vee M_i \neq M_j$
    - $\forall i \in [x], \mathsf{Tok.Verify}(S_i, M_i, D, \mathsf{pk}) = 1$
```

```
Oracle $\mathcal{O}^{\mathsf{Sign}}(M, D)$:
  $\mathsf{cnt}_D \leftarrow \mathsf{cnt}_D + 1$
  Return $\mathsf{Tok.Sign}(M, D, \mathsf{pk}, \mathsf{sk})$.
```

Figure 1: Strong One-More Unforgeability (SOMUF) Game with Public Metadata.

2.1 Unforgeability

We start with the first cryptographic guarantee provided by tokens known as unforgeability. At a high level, unforgeability guarantees that no party is able to generate signatures that will correctly verify unless they have access to the private signing key. For our work, we will consider a modification of the standard definition of unforgeability for blind signatures introduced by Schröder and Unruh [28]. This standard definition is known as *strong one-more unforgeability* as it was a strengthening of previous definitions such as those introduced in [27, 12].

We modify the definition of unforgeability to enable public metadata in the following way. In the prior definitions (without public metadata), the adversary wins if it can construct $\ell + 1$ signatures using at most ℓ signing oracle queries. In our game to incorporate public metadata, the adversary succeeds if it is able to construct at least $\mathsf{cnt}_D + 1$ signatures for any choice of public metadata D using at most cnt_D signing oracle queries with public metadata D. For example, this covers the case even when an adversarial user can forge a signature for some metadata D without ever sending a signing oracle query with D.

The security game can be found in Figure 1 and we present the formal definition below. We present our definition using concrete parameters for adversarial advantage ϵ, adversary running time t and number of oracle queries ℓ. This enables us to prove concrete security bounds throughout our paper.

Definition 2 ((ϵ, t, ℓ)-Strong One-More Unforgeableability)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{SOMUF}}_{\mathsf{Tok},\mathcal{A}}$ in Figure 1. A token scheme* Tok *is (ϵ, t, ℓ)-strong one-more unforgeable if, for any adversary $\mathcal{A}$ that runs in probabilistic time t and makes at most ℓ signing queries, the following holds*

$$\Pr[\mathsf{G}^{\mathsf{SOMUF}}_{\mathsf{Tok},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

Difference with Previous Definition. A slightly different definition of unforgeability for public metadata was introduced in [32]. For this game, the adversary is able to make at most ℓ signing queries for any fixed metadata. Then, the adversary wins the game if they pick some metadata D and can create $\ell + 1$ valid signatures for D. In our work, we instead bound the total number of signing queries by ℓ and the adversary wins the game if they are able to generate $\mathsf{cnt}_D + 1$ valid signatures for any metadata D while making at most cnt_D oracle queries for metadata D. The reason we do this is to be able to derive concrete security bounds with respect to ℓ. Using the prior definitions with public metadata in [32], we are unable to derive an upper bound on the number of signing queries beyond the running time of the adversary as the adversary may continue to make ℓ oracle queries for new choices of public metadata D. In other words, we can only guarantee that the adversary executes a probabilistically polynomial number of queries when considering probabilistically polynomial time adversaries.

Furthermore, we note that our definition is no weaker than the prior definition. If any adversary can produce $\ell + 1$ signatures using at most ℓ oracle queries, then there must exist some public metadata D such that the adversary has more valid signatures than oracle signing queries. We point readers to Appendix B for more details and showing the equivalence between the prior definition and our current definition.

Multiple Signers. The above definition of unforgeability only considers a single signer whereas general settings would consider multiple signers. Therefore, it may be natural and important to extend the definition

```
Game $\mathsf{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A}}(\lambda, b)$:
Adversary $\mathcal{A}$ outputs $(\mathsf{pk}, M_0, M_1, D) \leftarrow \mathcal{A}(1^\lambda)$.
For $i \in \{0,1\}$:
  Challenger $\mathcal{C}$ computes $(B_i, \mathsf{st}_i) \leftarrow \mathsf{Tok.Blind}(\mathsf{pk}, M_i, D)$.
$\mathcal{C}$ samples uniformly random bit $b \leftarrow_R \{0,1\}$.
$\mathcal{A}$ receives $B_b, B_{1-b}$ and computes $S'_b, S'_{1-b}$.
For $i \in \{0,1\}$:
  $\mathcal{C}$ computes $S_i \leftarrow \mathsf{Tok.Finalize}(\mathsf{pk}, \mathsf{st}_i, S'_i, M_i, D)$.
If $\mathsf{Tok.Verify}(M_0, S_0, D, \mathsf{pk}) = 1$ and $\mathsf{Tok.Verify}(M_1, S_1, D, \mathsf{pk}) = 1$:
  $\mathcal{C}$ sends $(M_0, S_0)$, $(M_1, S_1)$ to $\mathcal{A}$.
Else:
  $\mathcal{C}$ sends $\perp$ to $\mathcal{A}$.
$\mathcal{A}$ outputs a bit $b'$.
```

Figure 2: Unlinkability (UNLINK) Game.

for multiple signers. However, it turns out that the single and multiple signers are equivalent up to some factors that depend only on the number of signers. Furthermore, the strong one-more unforgebility is identical in either the single or multiple signer settings for two-move protocols (as studied in this paper). So, it is sufficient to focus security proofs with respect to a single signer. For more details, we point readers to [25].

Anonymous vs. Non-Anonymous Tokens. We will use the same definition for both anonymous and non-anonymous tokens and slightly overload notation for convenience. For non-anonymous tokens, we will assume that the input to the signing oracle will be the plaintext message that needs to be signed. In the case of anonymous tokens, the input to the signing oracle will be the blinded (i.e., encrypted) message where the signing oracle will perform the signer's actions to manipulate the blinded message such that the querier can retrieve the signed message.

Verification Oracles. In past definitions, an oracle for verification was also provided to the adversary. Although, this is only needed for the designated verifier setting where only certain parties can perform verification. For the case of RSA (blind) signatures, verification may be performed to anyone with access to public keys (that is, RSA signatures enable public verifiability). In other words, any adversary can simply execute the verification algorithm without the usage of an oracle. Therefore, there is no need for a verification oracle when considering publicly verifiable schemes.

2.2 Unlinkability

The second important cryptographic guarantee is anonymity that is typically referred to formally as unlinkability. In an anonymous token scheme, it must be impossible to determine the blind signing request that was utilized to create any signature even from the view of a signer that views the signing interactions, the final signatures as well as the corresponding input message of each signature.

We use the definition of unlinkability following the work from [7] for single-round schemes that we focus on throughout our work. We modify the definitions to encompass public metadata. In this game-based definition, the adversary picks the public key, two messages M_0 and M_1 as well as its choice of public metadata D. The challenger will first blind both messages and randomly permute the blinded messages. The blinded messages are submitted to the adversary in the randomly permuted order that will return signatures on the blinded messages. Finally, the challenge finalizes the adversary's responses to obtain the final signatures for both messages that are given to the adversary according to the original message order. The goal of the adversary is to guess the correct permutation of the original blinded messages. This security game is formally presented in Figure 2 and we present our definition of unlinkability below:

Definition 3 ((ϵ, t)-Unlinkability)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A},q}$ in Figure 2. A token scheme* Tok *satisfies (ϵ, t)-unlinkability if, for any adversary $\mathcal{A}$ running in time t, the following*

holds:
$$\left|\Pr[\mathsf{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A}}(\lambda,0)=1]-\Pr[\mathsf{G}^{\mathsf{UNLINK}}_{\mathsf{Tok},\mathcal{A}}(\lambda,1)=1]\right|\le\epsilon.$$

Discussion about Public Metadata. In our definition, the adversary is able to choose any choice of public metadata. However, the same public metadata must be used for signing both messages. This is necessary as if the adversary may pick different public metadata for each message then it can trivially win the game just by attempting to verify signatures with the two different public metadata. In practice, this implies that anonymity or unlinkability only applies to all groups of messages signed with the same public metadata. For real-world applications, it is important that the possible choices of public metadata is not too large to ensure anonymity is maintained. In an extreme case, if each message is signed with a unique choice of public metadata, then there are no anonymity guarantees anymore.

Adversarial Signer. Our choice of unlinkability definitions requires anonymity even against adversarial signers. In particular, we note that the adversary is able to choose the public key arbitrarily. Furthermore, the adversary is free to compute signatures on the blinded messages arbitrarily. Therefore, this encompasses all malicious signers that try to compromise anonymity.

Necessity of Successful Verification. In our definition, we note that the adversary is only permitted to see the resulting signatures if both signatures successfully verified. This is necessary to rule out one naive strategy for the adversary where it would issue a valid signature in response to one of the queries but not the other. In order to rule out this strategy, the challenger allows the adversary to see the resulting signatures only if both of them verify. If one (or both) of the signatures do not verify, the adversary will have to guess the ordering without seeing the finalized signatures.

3 Prior RSA Assumptions

In this section, we outline prior RSA assumptions that are relevant to our work. In particular, we present the standard RSA assumption as well as one-more RSA inversion assumptions.

3.1 Strong RSA Modulus

We start by discussing the RSA group structure. In our work, we will focus on a special subset of RSA modulus that are strong RSA modulus. All strong RSA modulus $N = pq$ are the product of two safe primes that enables structure to the multiplicative group modulo $\phi(N)$.

Definition 4 (Strong RSA Modulus)**.** *An integer N is a strong RSA modulus $N = p\cdot q$ where each of p and q are distinct safe primes. In other words, $p = 2p' + 1$ and $q = 2q' + 1$ where all of p, p', q, q' are distinct prime numbers. Therefore, $\phi(N) = 4p'q'$.*

As a side note, a key property of strong RSA modulus it that it provides significant structure to the group $\mathbb{Z}^*_{\phi(N)}$. One observation that we will rely upon throughout out work is that a random odd number from the set $[1, 2^{\lambda-3}]$ will be co-prime to $\phi(N)$. This is because $\phi(N) = 4p'q'$. Therefore, any odd number strictly smaller than both p' and q' is co-prime to $\phi(N)$. Since each of p' and q' are at least $\lambda - 2$ bits as p and q are λ bits, any odd number no larger than $2^{\lambda-3}$ is co-prime to $\phi(N)$.

3.2 RSA Assumption

We denote the probabilistic efficient algorithm $\mathsf{Gen}(1^\lambda)$ that generates a RSA modulus N and secret parameters (p, q) that are the prime factors of N. We will denote generating a random modulus, the public exponent distribution and secret parameters by $(N, \mathcal{D}_N, p, q) \leftarrow_R \mathsf{Gen}(1^\lambda)$.

We move onto defining the RSA assumption. At a high level, the RSA assumption assumes that any probabilistically polynomial time (PPT) adversary $\mathcal{A}$ given a RSA modulus N and public exponent e coprime to $\phi(N)$ is unable to compute the e-th root modulo N for a random element $X \leftarrow_R \mathbb{Z}^*_N$ drawn from the

```
Game $\mathsf{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:
$(N, \mathcal{D}_N, (p, q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$
$e \leftarrow_R \mathcal{D}_N$
$X \leftarrow_R \mathbb{Z}_N^*$
$Y \leftarrow \mathcal{A}(N, e, X)$
**Return** 1 if and only if $Y^e = X \bmod N$.
```

Figure 3: RSA Game.

multiplicative group modulo N. In other words, the PPT adversary $\mathcal{A}$ is unable to compute $X^d \bmod N$ where $d = e^{-1} \bmod \phi(N)$. In general, the RSA assumption is a class of assumptions that is parameterized by some distribution $\mathcal{D}_N$ determining how to pick the public exponent e. Thus, the Gen algorithm also outputs a distribution $\mathcal{D}_N$ to sample e. Furthermore, for concrete security bounds that may be used for guiding practical implementations, we consider a more fine-grained version of the RSA assumption that is additionally parameterized by the running time t and advantage ϵ of the adversary. Throughout the rest of this paper, we will present all security with respect to concrete parameters. As a result, all our definitions will always have fine-grained parameters.

Definition 5 ((ϵ, t)-RSA Assumption)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 3. The (ϵ, t)-RSA assumption is true for Gen if for any PPT adversary $\mathcal{A}$ that runs in time t,*

$$\Pr[\mathsf{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

In our work, we will focus on the setting when Gen always outputs strong RSA modulus N that is the product of two safe primes. We do not use the safe primes for reasons related to strengthening the RSA assumption. Instead, we use safe primes to utilize the additional structure in the multiplicative group $\mathbb{Z}^*_{\phi(N)}$. In particular, we show that safe primes enable an efficient and simple method of hashing public metadata to a large subset of elements in $\mathbb{Z}^*_{\phi(N)}$ without requiring knowledge of $\phi(N)$. We point readers to Section 5.2 for more details on our usage of the strong RSA modulus. We present our definition of the RSA assumption restricted to strong RSA modulus below.

Definition 6 ((ϵ, t)-RSA Assumption for Strong Modulus)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 3. The (ϵ, t)-RSA assumption is true for Gen such that Gen only outputs strong RSA modulus and, if for any PPT adversary $\mathcal{A}$ that runs in time t,*

$$\Pr[\mathsf{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

As the RSA assumption for strong modulus considers only a subset of possible modulus, it seems natural that it is no weaker than the standard RSA assumption. We formally prove this relation in the next theorem that utilizes prime number densities and the conjectured densities of safe prime numbers. We only assume that Gen generates N as the product of two uniformly random λ-bit primes, which is common practice (see BoringSSL [2] for example). See Appendix A for the proof and further details.

Theorem 1. *If the (ϵ, t)-RSA Assumption (Definition 5) is true and Gen generates modulus N as the product of two uniformly random λ-bit primes, then the $(O(\epsilon \cdot \lambda^2), t)$-RSA Assumption for Strong Modulus (Definition 6) is also true.*

In our proofs, we will prove the unforgeability of our RSA (non-blind) signatures with public metadata using the RSA assumption for strong modulus. Using Theorem 1, one can then re-interpret all our results as assuming the standard RSA assumption.

Finally, we note that the above reduction works for all of the assumptions in our paper. In particular, we can interchangeably use various RSA assumptions for arbitrary modulus as well as strong modulus. Throughout the rest of our work, we will use these assumptions interchangeably at the costs of an $O(\epsilon\lambda^2)$ multiplicative factor to the security parameter.

Game $\mathsf{G}^{\mathsf{CT\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:	Oracle $\mathcal{O}^{\mathsf{RSA}}(X)$:	Oracle $\mathcal{O}^{\mathcal{X}}()$:
$(N, \mathcal{D}_N, (p, q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$ $e \leftarrow_R \mathcal{D}_N$ $\phi(N) \leftarrow (p-1)(q-1)$ $\mathsf{cnt} \leftarrow 0$, $\mathcal{S}_{\mathcal{X}} \leftarrow \emptyset$ $(X_i, Y_i)_{i \in [\mathsf{cnt}+1]} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{RSA}}, \mathcal{O}^{\mathcal{X}}}(N, e)$ **Return** 1 if and only if all the following hold: - $\forall i \neq j \in [\mathsf{cnt}+1], X_i \neq X_j$ - $\forall i \in [\mathsf{cnt}+1], X_i \in \mathcal{S}_{\mathcal{X}}$ - $\forall i \in [\mathsf{cnt}+1], Y_i^e = X_i \bmod N$	$\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$ $d \leftarrow e^{-1} \bmod \phi(N)$ $Y \leftarrow X^d \bmod N$ **Return** Y	$X \leftarrow_R \mathbb{Z}_N^*$ $\mathcal{S}_{\mathcal{X}} \leftarrow \mathcal{S}_{\mathcal{X}} \cup \{X\}$ **Return** X

Figure 4: Chosen-Target RSA Inversion Game.

Discussion about $\mathcal{D}_N$. In practical implementations, the distribution $\mathcal{D}_N$ is typically a singleton fixed for all modulus N. The most common public exponents used in RSA implementations are 3 and 65537. On the other hand, theoretical works will typically consider $\mathcal{D}_N$ to be non-trivial distributions such as uniform from $\mathbb{Z}^*_{\phi(N)}$ or some subset of $\mathbb{Z}^*_{\phi(N)}$ (such as [17]). In our work, we will prove our construction to be secure for generic $\mathcal{D}_N$ with minimal assumptions. In general, we will only restrict the output size to be at most $e_{\mathbf{max}}$. In other words, $\mathcal{D}_N$ will pick a random element from $[1, e_{\mathbf{max}}]$ that is co-prime with $\phi(N)$. For our practical implementations, we fix $\mathcal{D}_N$ to be the small commonly used exponent as done in prior RSA implementations.

Discussion about Gen Trapdoor. In our definition of Gen, the trapdoor output is essentially two prime factors p and q of the RSA modulus N. Prior works defined Gen to return the decryption exponent $d = e^{-1} \bmod \phi(N)$ as opposed to the prime factorization of N. We note the two outputs are equivalent. Given the prime factors p and q of N, one can easily compute $\phi(N)$ and thus invert e modulo $\phi(N)$. In fact, we could also choose to define Gen to simply return $\phi(N)$ that is sufficient for our work. On the other hand, it is well known that one can also compute the prime factors of N using only d (see [17] for example). Therefore, the definitions are equivalent and we choose to return p and q for convenience.

3.3 One-More RSA Inversion Assumption

Security of RSA blind signature schemes have been proven by using of a class of computational problems known as one-more RSA inversion problems introduced by Bellare, Namprempre, Pointcheval and Semanko [12]. In these problems, the adversary is given access to a decryption oracle $\mathcal{O}^{\mathsf{RSA}}$ and a challenge oracle $\mathcal{O}^{\mathcal{X}}$. The decryption oracle takes as input $X \in \mathbb{Z}_N^*$ and returns $\mathcal{O}^{\mathsf{RSA}}(X) = Y$ where $Y^e = X \bmod N$. The challenge oracle $\mathcal{O}^{\mathcal{X}}$ will return random elements from $\mathbb{Z}_N^*$ as random challenge targets. The adversary only succeeds if it inverts $\ell + 1$ challenge targets while making at most ℓ queries to the decryption oracle $\mathcal{O}^{\mathsf{RSA}}$. That is, it computes at least one more RSA decryption than the number of oracle queries. We define chosen-target RSA inversion game in Figure 4 along with the following definition that were both introduced in [12]. In this definition, *chosen-target* implies that the adversary may choose to invert any of the targets output by the oracle $\mathcal{O}^{\mathcal{X}}$.

Definition 7 ((ϵ, t, ℓ)-Chosen-Target RSA Inversion Assumption)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{CT\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 4. The (ϵ, t, ℓ)-chosen-target RSA inversion assumption is true for* Gen *if, for any adversary $\mathcal{A}$ that runs in time t and makes at most ℓ decryption queries, the following holds*

$$\Pr[\mathsf{G}^{\mathsf{CT\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

Discussion about Challenge Oracle. In the formulation of the chosen-target RSA inversion game, the adversary has access to a target oracle $\mathcal{O}^{\mathcal{X}}$ that it can use to get a set of potential targets to invert. Another definition may be to suppose that the set of possible targets is provided to the adversary as input instead. It turns out that the two definitions are equivalent as shown in [12]. Throughout the rest of the work, we will consider this formulation where the adversary will have access to a challenge oracle.

```
Game $\mathsf{G}^{\mathsf{ME\text{-}RSA}}_{\mathsf{Gen},\mathcal{A},\ell}(\lambda)$:
$(N, \mathcal{D}_N, (p, q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$
$(e_1, \ldots, e_\ell) \leftarrow_R \mathcal{D}_N$
$X \leftarrow_R \mathbb{Z}_N^*$
$(i, Y) \leftarrow \mathcal{A}(N, e_1, \ldots, e_\ell, X)$
**Return** 1 if and only if $i \in [l]$ and $Y^{e_i} = X \bmod N$.
```

Figure 5: Multi-Exponent RSA Game. All differences with the RSA game, $\mathsf{G}^{\mathsf{RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$, are highlighted in blue.

4 RSA Assumptions with Multiple Exponents

In this section, we introduce and explore various generalizations of the RSA assumption with multiple exponents. Afterwards, we further extend them to one-more variants (following the definitional extensions for the RSA assumption in [12]). To our knowledge, we are unaware of prior works that formally defined multi-exponent variants of the RSA assumption. In our work, we will explore various natural generalizations and analyze them carefully.

4.1 Multi-Exponent RSA Assumption

We start by extending the standard RSA assumption from one challenge exponent e to a set of ℓ exponents, $e_1, \ldots, e_\ell$. The adversary wins the game if it can decrypt a random target for any of the ℓ exponents provided in the challenge. We formally present this game in Figure 5 that we denote as the multi-exponent RSA game.

Definition 8 ((ϵ, t, ℓ)-Multi-Exponent RSA Assumption)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{ME\text{-}RSA}}_{\mathsf{Gen},\mathcal{A},\ell}(\lambda)$ in Figure 5. The (ϵ, t, ℓ)-multi-exponent RSA assumption is true for Gen, if for any adversary $\mathcal{A}$ that runs in time t, the following holds*

$$\Pr[\mathsf{G}^{\mathsf{ME\text{-}RSA}}_{\mathsf{Gen},\mathcal{A},\ell}(\lambda) = 1] \leq \epsilon.$$

As this provides more flexibility for the adversary, it is a stronger assumption than the standard RSA assumption with a single exponent. However, we can show that the two assumptions differ by at most an additive factor that is exponentially smaller in λ for reasonable choices of $e_{\mathbf{max}}$.

Theorem 2. *If the (ϵ, t)-RSA assumption for strong modulus (Definition 6) is true with exponents from $[1, e_{\mathbf{max}}]$, then the $(\epsilon + \ell \cdot (e_{\mathbf{max}}/e'_{\mathbf{max}}), t - O(\ell), \ell)$-multi-exponent RSA assumption (Definition 8) is true with exponents from $[1, e'_{\mathbf{max}}]$ where $e_{\mathbf{max}} < e'_{\mathbf{max}} \leq 2^{\lambda-2}$.*

Proof. Towards a contradiction, suppose that there exists an adversary $\mathcal{A}'$ that wins the multi-exponent RSA game with probability ϵ' with running time t' when given ℓ challenge exponents. We build an adversary $\mathcal{A}$ for the RSA game that wins with probability ϵ' with the running time $t = t' + O(\ell)$.

Recall that $\mathcal{A}$ is given an exponent e and random target X and must produce Y satisfying $Y^e = X \bmod N$. To do this, $\mathcal{A}$ will pick ℓ random odd numbers from the set $[1, e'_{\mathbf{max}}/e]$ that we denote as $z_1, \ldots, z_\ell$. We choose the ℓ challenge exponents as $e_i = e \cdot z_i$ for all $i \in [\ell]$. As we assume N is a strong RSA modulus and the product of two λ-bit primes, we know that $\phi(N) = 4p'q'$ for some primes p' and q' of length at least $\lambda - 2$. Therefore, odd numbers from $[1, 2^{\lambda-2}]$ will be co-prime with $\phi(N)$ and, thus, valid exponents. As each z_i are chosen such that $e_i = e \cdot z_i \leq e'_{\mathbf{max}} \leq 2^{\lambda-2}$, we know that each e_i is a valid exponent. Finally, $\mathcal{A}$ passes the modulus N, the ℓ exponents as well as the random target X to $\mathcal{A}'(N, e_1, \ldots, e_\ell, X)$.

First, we present a hybrid game between the real and simulated view of $\mathcal{A}'$. Note that $\mathcal{A}'$ receives random exponents from $[1, e'_{\mathbf{max}}]$ in the real game. In the simulated game, $\mathcal{A}'$ receives random exponents from $[e_{\mathbf{max}}, e'_{\mathbf{max}}]$. As each exponent is generated at random from $[1, e'_{\mathbf{max}}]$, we note that $\mathcal{A}'$ can only distinguish the two views with probability at most $\ell \cdot ((e_{\mathbf{max}}/2)/(e'_{\mathbf{max}}/2)) = \ell \cdot (e_{\mathbf{max}}/e'_{\mathbf{max}})$.

Next, we assume that $\mathcal{A}'$ cannot distinguish between the two views. In the simulated view, $\mathcal{A}'$ receives ℓ challenge exponents from $[e_{\mathbf{max}}, e'_{\mathbf{max}}]$ that are co-prime to $\phi(N)$. Note that $\mathcal{A}'$ returns i and a valid

Game $\mathsf{G}^{\mathsf{CT\text{-}RE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:	Oracle $\mathcal{O}^{\mathsf{RSA}}(X,e)$:	Oracle $\mathcal{O}^{\mathcal{X}}()$:	Oracle $\mathcal{O}^{\mathsf{exp}}()$:
$(N,\mathcal{D}_N,(p,q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$	If $e \notin \mathcal{S}_{\mathsf{exp}}$:	$X \leftarrow_R \mathbb{Z}_N^*$	$e \leftarrow_R \mathcal{D}_N$
$\mathcal{S}_{\mathcal{X}} \leftarrow \emptyset$, $\mathcal{S}_{\mathsf{exp}} \leftarrow \emptyset$	**Return** $\perp$	$\mathcal{S}_{\mathcal{X}} \leftarrow \mathcal{S}_{\mathcal{X}} \cup \{X\}$	$\mathcal{S}_{\mathsf{exp}} \leftarrow \mathcal{S}_{\mathsf{exp}} \cup \{e\}$
$\phi(N) \leftarrow (p-1)(q-1)$	$\mathsf{cnt}_e \leftarrow \mathsf{cnt}_e + 1$	**Return** X	**Return** e
$\mathsf{cnt}_e \leftarrow 0, \forall e \in \mathbb{Z}^*_{\phi(N)}$	$d \leftarrow e^{-1} \bmod \phi(N)$		
$e,\ (X_i,Y_i)_{i\in[x]} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{RSA}},\mathcal{O}^{\mathcal{X}},\mathcal{O}^{\mathsf{exp}}}(N)$	$Y \leftarrow X^d \mod N$		
Return 1 if and only if all the following hold:	**Return** Y		
- $\mathsf{cnt}_e < x$			
- $e \in \mathcal{S}_{\mathsf{exp}}$			
- $\forall i \neq j \in [x], X_i \neq X_j$			
- $\forall i \in [x], X_i \in \mathcal{S}_{\mathcal{X}}$			
- $\forall i \in [x], Y_i^e = X_i \mod N$			

Figure 6: Chosen-Target, Restricted-Exponent RSA Inversion Game. All differences with the chosen-target RSA inversion game, $\mathsf{G}^{\mathsf{CT\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$, are highlighted in blue.

decryption of X under e_i with probability ϵ'. In other words, $\mathcal{A}'$ outputs Y satisfying $Y = X^{1/e_i} = X^{1/(e\cdot z_i)}$. Therefore, $\mathcal{A}$ can output $Y^{z_i} = X^{1/e}$ to win the RSA game. So, $\mathcal{A}$ wins this game with probability ϵ' assuming the simulated and real views are indistinguishable. Altogether, $\mathcal{A}$ wins the game with probability $\epsilon' - \ell \cdot (e_{\mathbf{max}}/e'_{\mathbf{max}})$.

For the running time, note that $\mathcal{A}$ required an additional $O(\ell)$ time to generate random exponents. If $\mathcal{A}'$ runs in time t', then $\mathcal{A}$ runs in time $t' + O(\ell)$. □

Note, we can set $e_{\mathbf{max}} \leq 2^{\lambda/2}$ to obtain that the difference between the standard and multi-exponent RSA game is at most an additive factor of $2^{-\lambda/2+\log \ell+2}$.

We build upon the above ideas to obtain our one-more extension of the multi-exponent RSA assumption that we will use to prove security of our blind signature protocol.

4.2 One-More Multi-Exponent RSA Assumption

We start from the chosen-target variant of the one-more RSA assumption from [12] (we refer readers back to Figure 4 for full details). We will modify this game to obtain a one-more variant of the multi-exponent RSA assumption from the prior section. As the first step, we will also create an exponent oracle $\mathcal{O}^{\mathsf{exp}}$ that will output challenge exponents (analogous to the oracle $\mathcal{O}^{\mathcal{X}}$ for challenge targets). Next, we generalize the decryption oracle, $\mathcal{O}^{\mathsf{RSA}}$, to receive both an element X and an exponent e. However, $\mathcal{O}^{\mathsf{RSA}}$ makes the restriction that the input exponent e must be a challenge exponent output by $\mathcal{O}^{\mathsf{exp}}$. In other words, the decryption oracle only works for the exponents output by $\mathcal{O}^{\mathsf{exp}}$.

This immediately leads to the first natural definition of a one-more multi-exponent RSA assumption where an adversary makes at most cnt_e decryption oracle queries for some exponent e and must output $\mathsf{cnt}_e + 1$ valid decryptions of the form $(X_i, Y_i)_{i\in[\mathsf{cnt}_e+1]}$ satisfying that $Y_i^e = X_i \bmod N$ where e must be some challenge exponent output by $\mathcal{O}^{\mathsf{exp}}$. We keep the same restriction as the one-more RSA assumption that the targets X_i must be valid outputs from the message oracle $\mathcal{O}^{\mathcal{X}}$ (satisfying the chosen-target property).

We formally define this game in Figure 6 that we denote as the chosen-target, restricted-exponent RSA inversion game. In this definition, we use *restricted-exponent* to denote the fact that the decryption oracle $\mathcal{O}^{\mathsf{RSA}}$ restricts the adversary to only pick exponents output by $\mathcal{O}^{\mathsf{exp}}$.

Definition 9 ((ϵ, t, ℓ)-Chosen-Target, Restricted-Exponent RSA Inversion Assumption)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{CT\text{-}RE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 6. The (ϵ, t, ℓ)-chosen-target, restricted-exponent RSA inversion assumption is true for* Gen*, if for any adversary $\mathcal{A}$ that runs in time t and makes at most ℓ decryption queries, the following holds*

$$\Pr[\mathsf{G}^{\mathsf{CT\text{-}RE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

```
Game $\mathsf{G}^{\mathsf{SRSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:
$(N, (p, q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$
$X \leftarrow_R \mathbb{Z}_N^*$
$(e, Y) \leftarrow \mathcal{A}(\mathsf{crs}, N, X)$
Return 1 if and only if $e \geq 3$ and $Y^e = X \bmod N$.
```

Figure 7: Strong RSA Game.

Relation to Chosen-Target RSA Inversion Assumption. In the prior section, we showed that extending the standard RSA assumption from one exponent to ℓ exponents only affected the adversarial advantage by an additive factor exponentially small in λ. In other words, the assumptions are essentially identical when considering polynomial time adversaries.

One could attempt to try and prove a similar equivalence between the chosen-target, restricted-exponent RSA inversion assumption (Definition 9) and the original chosen-target RSA inversion assumption (Definition 7) from [12]. Clearly, the chosen-target, restricted-exponent variant is a no weaker assumption than the original chosen-target variant. Recall that, in the proof of Theorem 2, it was required to generate fake challenge exponents co-prime to $\phi(N)$. This can be easily done when N is a strong RSA modulus. For this reduction, we would need to additionally be able to produce decryptions with respect to the fake challenge exponents. Unfortunately, this seems quite challenging. For example, suppose an adversary tried to produce pairs (e', d') such that $e'd' = 1 \bmod \phi(N)$. Then, the adversary could submit e' as a challenge exponent and use d' to answer decryption queries. It is well known that producing a pair (e', d') satisfying this property is equivalent to factoring N and, thus, breaking any RSA assumption immediately. Therefore, this seems like a very challenging task for an adversary without knowledge of $\phi(N)$.

We leave it as an open problem to determine the relationship between the chosen-target, restricted-exponent inversion assumption and the original chosen-target inversion assumption.

4.3 Connections to Strong RSA Assumption

Finally, we make connections between the above multi-exponent RSA assumptions and the strong RSA assumption introduced in [11, 21]. We emphasize that the assumptions in this section that will not be used for proving security of any of our protocols. We will only rely on the multi-exponent variants from Section 4.2 for our security proofs. Instead, we perform an exploration to see the connections between the assumptions from our prior section and its relation to the strong RSA assumption.

Strong RSA Assumption. In comparison with the standard RSA assumption, the strong RSA assumption provides more flexibility to the adversary to choose its own public exponent. The strong RSA problem states that the original RSA problem is intractable even when $\mathcal{A}$ is allowed to choose the public exponent $e \geq 3$. More specifically, given a RSA modulus N, and a random target X, it is infeasible to find any pair (Y, e) such that $X = Y^e \bmod N$ and $e \geq 3$. We provide the formal definition of the strong RSA assumption below:

Definition 10 ((ϵ, t)-Strong RSA Assumption)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{SRSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 7. The (ϵ, t)-strong RSA assumption is true for* Gen*, if for any PPT adversary $\mathcal{A}$ that runs in time t, the following holds*

$$\Pr[\mathsf{G}^{\mathsf{SRSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

Extending towards One-More Strong RSA Type Assumptions. A natural next step is to extend the strong RSA assumption using the one-more style definitions. To our knowledge, we are unaware of prior work that has formally defined this notion. We will present two natural definitions that seem to be reasonable. We show that one such definition seems plausible while the other can be broken by an adversary.

Before presenting these definitions, we quickly overview why the prior chosen-target, restricted-exponent RSA assumption (Definition 9) from Section 4.2 seems significantly weaker. Recall that, in the strong RSA assumption, the adversary is able to choose any exponent $e \geq 3$. In contrast, for the chosen-target,

Game $\mathsf{G}^{\text{CT-CE-RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:	Oracle $\mathcal{O}^{\mathsf{RSA}}(X, e)$:	Oracle $\mathcal{O}^{\mathcal{X}}()$:	Oracle $\mathcal{O}^{\mathsf{exp}}()$:
$(N,\, \mathcal{D}_N,\, (p, q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$	If $e \notin \mathbb{Z}^*_{\phi(N)}$:	$X \leftarrow_R \mathbb{Z}^*_N$	$e \leftarrow_R \mathcal{D}_N$
$\mathcal{S}_\mathcal{X} \leftarrow \emptyset$, $\mathcal{S}_{\mathsf{exp}} \leftarrow \emptyset$	**Return** $\perp$	$\mathcal{S}_\mathcal{X} \leftarrow \mathcal{S}_\mathcal{X} \cup \{X\}$	$\mathcal{S}_{\mathsf{exp}} \leftarrow \mathcal{S}_{\mathsf{exp}} \cup \{e\}$
$\phi(N) \leftarrow (p-1)(q-1)$	$\mathsf{cnt}_e \leftarrow \mathsf{cnt}_e + 1$	**Return** X	**Return** e
$\mathsf{cnt}_e \leftarrow 0, \forall e \in \mathbb{Z}^*_{\phi(N)}$	$d \leftarrow e^{-1} \bmod \phi(N)$		
$e, (X_i, Y_i)_{i\in[x]} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{RSA}}, \mathcal{O}^{\mathcal{X}}, \mathcal{O}^{e}}(N)$	$Y \leftarrow X^d \mod N$		
Return 1 if and only if all the following hold:	**Return** Y		
- $\mathsf{cnt}_e < x$			
- $e \in \mathcal{S}_{\mathsf{exp}}$			
- $\forall i \neq j \in [x], X_i \neq X_j$			
- $\forall i \in [x], X_i \in \mathcal{S}_\mathcal{X}$			
- $\forall i \in [x], Y_i^e = X_i \bmod N$			

Figure 8: Chosen-Target, Chosen-Exponent RSA Inversion Game. All differences with the chosen-target, restricted-exponent RSA inversion game, $\mathsf{G}^{\text{CT-RE-RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$, are highlighted in blue.

restricted-exponent game, $\mathsf{G}^{\text{CT-RE-RSA}}_{\mathsf{Gen},\mathcal{A},\ell}(\lambda)$, the adversary is significantly restricted in the exponents of choice. The chosen-target, restricted-exponent definition places two requirements on the adversaries that seem more stringent than the strong RSA assumption. First, the output exponent must be one of the outputs of the challenge oracle $\mathcal{O}^{\mathsf{exp}}$. Secondly, the adversary is not able to submit any decryption requests for exponents except for those output by $\mathcal{O}^{\mathsf{exp}}$. Therefore, in our opinion, the chosen-target, restricted-exponent assumption is weaker and not a truly natural extension of the strong RSA assumption. Intuitively, the chosen-target, restricted-exponent assumption seems significantly weaker than any true one-more strong RSA assumption.

Next, we will consider assumptions that are (seemingly) stronger than the chosen-target, restricted-exponent game that get closer to the spirit of an one-more strong RSA assumption.

One-More RSA with Chosen Exponents. As our first attempt to consider stronger assumptions, we will consider a new game where the decryption oracle, $\mathcal{O}^{\mathsf{RSA}}$, allows the adversary to submit arbitrary exponents for decryption. In particular, we no longer restrict queries to the decryption oracle to consist of exponents output by the challenge oracle $\mathcal{O}^{\mathsf{exp}}$. As a technical detail, we do require queried exponents to be co-prime to $\phi(N)$ to enable decryption. In the case that the input exponent is not co-prime to $\phi(N)$, $\mathcal{O}^{\mathsf{RSA}}$ will output $\perp$. Note, we will still require the adversary to output exponents from $\mathcal{O}^{\mathsf{exp}}$ to win the game though. Clearly, this is a stronger assumption compared to the chosen-target, restricted-exponent assumption (Definition 9) as the adversary has significantly more freedom when choosing decryption oracle queries.

We formally present this game in Figure 8 that we denote as the chosen-target, chosen-exponent RSA inversion game. We use the notion *chosen-exponent* to denote that the adversary is now free to pick any exponents to submit to the decryption oracle $\mathcal{O}^{\mathsf{RSA}}$.

Definition 11 ((ϵ, t, ℓ)-Chosen-Target, Chosen-Exponent RSA Inversion Assumption)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\text{CT-CE-RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$ in Figure 8. The (ϵ, t, ℓ)-chosen-target, chosen-exponent RSA inversion assumption is true for* Gen *if, for any adversary $\mathcal{A}$ that runs in time t and makes at most ℓ decryption queries, the following holds*

$$\Pr[\mathsf{G}^{\text{CT-CE-RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda) = 1] \leq \epsilon.$$

While this seems like a reasonable assumption, it still limits the adversary's abilities to pick exponents e. In particular, the adversary is forced to use one of the outputs of the challenge oracle $\mathcal{O}^{\mathsf{exp}}$. In contrast, the strong RSA assumption enables the adversry to pick any $e \geq 3$ and produce a decryption with respect to e.

One-More RSA with Arbitrary Exponents. As the final assumption, we remove the above restriction and enable the adversary to pick arbitrary exponents identical to the strong RSA assumption. We no longer provide a challenge exponent oracle, $\mathcal{O}^{\mathsf{exp}}$, to the adversary. Instead, the adversary is free to choose any exponent $e \geq 3$ and produce a decryption with respect to e. We formally present this game in Figure 9

Game $\mathsf{G}^{\mathsf{CT\text{-}AE\text{-}RSA}}_{\mathsf{Gen},\mathcal{A}}(\lambda)$:	Oracle $\mathcal{O}^{\mathsf{RSA}}(X, e)$:	Oracle $\mathcal{O}^{\mathcal{X}}()$:
$(N, (p, q)) \leftarrow_R \mathsf{Gen}(1^\lambda)$	If $e \notin \mathbb{Z}^*_{\phi(N)}$:	$X \leftarrow_R \mathbb{Z}^*_N$
$\mathcal{S}_{\mathcal{X}} \leftarrow \emptyset$	**Return** $\perp$	$\mathcal{S}_{\mathcal{X}} \leftarrow \mathcal{S}_{\mathcal{X}} \cup \{X\}$
$\phi(N) \leftarrow (p-1)(q-1)$	$\mathsf{cnt}_e \leftarrow \mathsf{cnt}_e + 1$	**Return** X
$\mathsf{cnt}_e \leftarrow 0, \forall e \in \mathbb{Z}^*_{\phi(N)}$	$d \leftarrow e^{-1} \bmod \phi(N)$	
$e, (X_i, Y_i)_{i \in [x]} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{RSA}}, \mathcal{O}^{\mathcal{X}}}(N)$	$Y \leftarrow X^d \mod N$	
Return 1 if and only if all the following hold:	**Return** Y	
- $\mathsf{cnt}_e < x$		
- $\forall i \neq j \in [x], X_i \neq X_j$		
- $\forall i \in [x], X_i \in \mathcal{S}_{\mathcal{X}}$		
- $\forall i \in [x], Y_i^e = X_i \mod N$		

Figure 9: Chosen-Target, Arbitrary-Exponent RSA Inversion Game.

that we denote as the chosen-target, arbitrary-exponent RSA inversion game. We denote this game with the notion of *arbitrary-exponent* as the adversary is free to output any exponent $e \geq 3$.

While this seems like a natural extension, we can show that arbitrary exponents provide far too much power to an adversary. In fact, we can present a simple polynomial time adversary that can break this game.

Theorem 3. *There exists a polynomial time adversary $\mathcal{A}$ that wins the chosen-target, arbitrary-exponent RSA inversion game in Figure 9.*

Proof. We present a simple adversary $\mathcal{A}$ as follows. First, $\mathcal{A}$ gets a target $X \leftarrow \mathcal{O}^{\mathcal{X}}()$. Next, the $\mathcal{A}$ picks two different numbers e and e' that are co-prime to $\phi(N)$. To find these, $\mathcal{A}$ can submit decryption oracles with random odd numbers as exponents that only returns $\perp$ when the submitted exponent is not co-prime to $\phi(N)$. Without loss of generality, let $e' > e$. The adversary calls the decryption oracle to receive $Y \leftarrow \mathcal{O}^{\mathsf{RSA}}(X^{e'}, e \cdot e')$. Then, we note that $Y = (X^{e'})^{1/(e \cdot e')} = X^{1/e}$. Finally, $\mathcal{A}$ outputs $e, (X, Y)$ and wins the game as the adversary never made a single decryption oracle call for e. □

Interestingly, the ability for the adversary to choose arbitrary exponents without a challenge exponent oracle $\mathcal{O}^{\mathsf{exp}}$ enables far too much adversarial power. The same reasoning was used in prior RSA assumptions where the target was given at random to the adversary. If the adversary is able to pick arbitrary messages to decrypt, then the adversary can easily break prior RSA assumptions as well. While this does not occur for exponents in the standard assumptions without decryption oracles, it ends up being the case once you provide decryption oracles to the adversary. Therefore, the challenge exponent oracle $\mathcal{O}^{\mathsf{exp}}$ seems necessary for any of these one-more RSA assumptions.

Inability to Extend Attack. One could try to extend these attacks to our other assumptions such as the chosen-target, chosen-exponent assumption (Definition 11) or the chosen-target, restricted-exponent assumption (Definition 9). In the attack, the adversary is required to find two exponents e and e' such that $e \mid e'$. In the chosen-exponent game, it is unlikely the adversary will ever receive such a pair of exponents from the oracle $\mathcal{O}^{\mathsf{exp}}$. For example, if N is a strong RSA modulus such that $N = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, then $\phi(N) = 4p'q'$. Assuming that p' and q' are sufficiently large, then it is very unlikely for an adversary to receive to random elements e, e' from $\mathbb{Z}^*_{\phi(N)}$ (via the exponent oracle, $\mathcal{O}^{\mathsf{exp}}$) satisfying $e \mid e'$. Therefore, we were unable to extend this attack to any of the prior games due to the requirement that the adversary must use exponents output by oracle $\mathcal{O}^{\mathsf{exp}}$.

5 RSA Signatures with Public Metadata

In this section, we start by presenting a protocol for RSA (non-blind) signatures with public metadata. The goal is to show that our protocol augmentation for public metadata does not degrade concrete security in any meaningful way compared to the original protocol without public metadata.

5.1 Preliminaries

Before we present our protocol, we start by presenting various preliminaries and building blocks that we will utilize throughout our protocols. We will use these primitives in our RSA blind signature protocol in Section 6 as well.

Unique Concatenation. Throughout our work, we will utilize concatenation of multiple values where each unique tuple should result in a unique concatenation. Formally, we may consider tuples (A, B) such that the concatenation $A \mid\mid B$ must be unique. That is, for any pair of tuples, (A, B) and (A', B'), it must be that $A \mid\mid B$ is identical to $A' \mid\mid B'$ if and only if $A = A'$ and $B = B'$. This could also extend to tuples of size larger than two. A straightforward way to do this is to encode tuples (A, B) as $|A| \mid\mid A \mid\mid B$ where $|A|$ is the length of A stored in some fixed-length integer (such as 64 bits). This may be extended to longer tuples naturally as $(X_1, X_2, \ldots, X_n)$ may be concatenated as $|X_1| \mid\mid |X_2| \mid\mid \ldots \mid\mid |X_{n-1}| \mid\mid X_1 \mid\mid X_2 \mid\mid \ldots \mid\mid X_n$. For the rest of the work, we will assume that when we concatenate any tuple of values $X_1 \mid\mid X_2 \mid\mid \ldots \mid\mid X_n$, we are using the unique version with length prepending.

Message Encoding. In our protocols, we will utilize standard encoding algorithms for messages used in RSA signatures. Formally, we will assume that the message encoding consists of two algorithms: $H_{\mathcal{M}}$ and $\mathsf{Verify}_{\mathcal{M}}$. The hash function $H_{\mathcal{M}}(X) : \{0,1\}^* \to \mathbb{Z}_N^*$ maps strings to elements in the multiplicative group modulo the RSA modulus N. The verification algorithm $\mathsf{Verify}_{\mathcal{M}}(X, Y) : \{0,1\}^* \times \mathbb{Z}_N^* \to \{0,1\}$ checks and returns 1 if and only if Y corresponds to the output of $H_{\mathcal{M}}(X)$.

In our work, we will consider two different encodings: full domain hash (FDH) and probabilistic signature scheme (PSS). At a high level, FDH is a determinsitic encoding algorithm that maps each message to a random element. In contrast, PSS is a randomized encoding algorithm that will map the same message to different outputs when evoked multiple times. Bellare and Rogaway [13] showed that the concrete security of PSS encodings was stronger than FDH encodings when applied to standard RSA signatures (without public metadata). In our implementations, we use PSS encodings due to its superior concrete security and randomized properties. Furthermore, we wish to align our implementation with current IRTF specifications for RSA blind signatures [20] that do not consider public metadata. The version of PSS encodings that we rely upon are those from prior specifications [26] that essentially uses a variant of PSS encodings with message recovery presented in [13].

For our proofs, we will prove security of our schemes assuming the usage of the FDH encoding. We chose this approach as to not over-complicate our proof with PSS encoding techniques that are directly borrowed from prior work [13]. Instead, we wish to focus our proof on the new techniques required to prove security with respect to the additional public metadata functionality. Although, one can directly apply the proof techniques for PSS encodings from [13] to our security proofs. As we will use FDH, we present it formally.

Definition 12 (Full Domain Hash)**.** *The full domain hash (FDH) message encoding consists of two deterministic algorithms $H_{\mathcal{M}} : \{0,1\}^* \to \mathbb{Z}_N^*$ and $\mathsf{Verify}_{\mathcal{M}} : \{0,1\}^* \times \mathbb{Z}_N^* \to \{0,1\}$ as follows:*

- *For each string X, $H_{\mathcal{M}}(X)$ is a random element from $\mathbb{Z}_N^*$.*
- *For a string X and $Y \in \mathbb{Z}_N^*$, $\mathsf{Verify}_{\mathcal{M}}(X, Y) = 1$ if and only if $Y = H_{\mathcal{M}}(X)$.*

Instantiations of Hash Functions. Throughout the rest of this section and our proofs, we will assume that all hash functions are modeled as random oracles unless otherwise specified. In practice, we will instantiate these hash function using a variety of different approaches depending on prior standards or the specific requirements. We defer to prior specifications including [20, 10] for recommendations on specific instantiations of each hash function.

5.2 Protocol for RSA Signatures with Public Metadata

In this section, we formally present the protocol for RSA signatures with public metadata. Our goal of starting with the non-blind version is to show that our public metadata augmentation does not degrade the concrete security compared to the standard RSA signature scheme without public metadata. In particular, we will show that concrete security degrades by an additive factor of $2^{O(-\lambda)}$ where λ is the length of the prime factors of the strong RSA modulus. In practice, this is essentially negligible. We believe this provides evidence that our new protocol with public metadata augmentation has similar security as the prior RSA blind signature without public metadata.

Setup. For the reader's convenience, we quickly summarize our setup. We will denote the strong RSA modulus by N of bit length 2λ such that the two safe primes p, q (factors of N) are both of bit length λ. The public key will be $\mathsf{pk} \leftarrow N$ which is the strong RSA modulus. The private key will be $\mathsf{sk} \leftarrow \phi(N)$. As $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, we know that both p' and q' are also primes. Therefore, $\phi(N) = (p-1) \cdot (q-1) = 4p'q'$.

Public Metadata Hash Function. Next, we define our hash function that is used to map public metadata to values in the group of RSA exponents $\mathbb{Z}^*_{\phi(N)}$. Recall that these are all elements that are invertible modulo $\phi(N)$. This is a straightforward task if $\phi(N)$ was publicly known by all parties. However, it is critical that $\phi(N)$ is hidden to all parties except the signer. Therefore, we must come up a way to perform this mapping for users that do not know the value of $\phi(N)$. We present a simple way to do this below that relies on the structure of $\mathbb{Z}^*_{\phi(N)}$ and the fact that N is a strong RSA modulus.

$H_{\mathsf{MD}}(D)$: Let G be a hash function.

1. Compute $G(D)$ of length γ bits.
2. Return $1 \,||\, G(D)$. In other words, we are returning $2 \cdot G(D) + 1$.

In words, the above function simply returns a random odd number of length $\gamma + 1$ bits. Picking γ correctly, we can guarantee that the output of H_{MD} will always be smaller than the prime values p' and q' such that $\phi(N) = 4p'q'$. This turns out to be sufficient to guarantee that outputs of H_{MD} will always be elements from $\mathbb{Z}^*_{\phi(N)}$.

Lemma 1. *Suppose N is a strong RSA modulus such that $N = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes each of length λ and $\gamma \leq \lambda - 4$. Then, for all $D \in \{0,1\}^*$, $H_{\mathsf{MD}}(D)$ is co-prime to $\phi(N)$.*

Proof. Fix any $x \in \{0,1\}^*$. First, we note that the output of $H_{\mathsf{MD}}(x)$ is always odd. Therefore, we know that if $H_{\mathsf{MD}}(x)$ is not co-prime to $\phi(N)$, then it must be that either $p' \mid H_{\mathsf{MD}}(x)$ or $q' \mid H_{\mathsf{MD}}(x)$. We know that the bit length of p' and q' is at least $\lambda - 2$ given that $p = 2p' + 1$ and $q = 2q' + 1$ and both p and q are λ bit long primes. Note that the bit length of $H_{\mathsf{MD}}(x)$ is $\gamma + 1 \leq \lambda - 3$. As p and q are bit length λ, we know that both p' and q' have bit length at least $\lambda - 2$. Thus, we know that $H_{\mathsf{MD}}(x) < p'$ and $H_{\mathsf{MD}}(x) < q'$. Therefore, $H_{\mathsf{MD}}(x)$ is always co-prime to $\phi(N)$. □

RSA Signatures with Public Metadata. Using the public metadata hash function H_{MD}, we are ready to present a description of our RSA signatures with public metadata. We will assume that $H_{\mathcal{M}}$ and $\mathsf{Verify}_{\mathcal{M}}$ correspond to some message encoding algorithm (such as FDH in Definition 12). We note that the Setup algorithm is essentially identical to the standard RSA signatures without public metadata protocols. The signing algorithm is modified to use $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$ where D is the public metadata. As the signer knows $\phi(N)$, it can compute $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$ efficiently to produce a signature. The verification algorithm is identical when using e_{MD} as the public exponent. As we are considering non-blind signatures (i.e., non-anonymous tokens), the Blind and $\mathsf{Finalize}$ algorithms are trivial and, thus, omitted (see Section 2 for definitions). We present the formal algorithms for all of the necessary algorithms below.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Setup}(1^\lambda)$:

1. Randomly generate $(N, p, q) \leftarrow_R \mathsf{Gen}(1^\lambda)$.
2. Compute $\phi(N) \leftarrow (p-1)(q-1)$.

3. Return $\mathsf{pk} \leftarrow N, \mathsf{sk} \leftarrow \phi(N)$.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Sign}(M, D, \mathsf{pk} \leftarrow N, \mathsf{sk} \leftarrow \phi(N))$:

1. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
2. Compute $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$.
3. Compute $M_{\mathsf{MD}} \leftarrow H_{\mathcal{M}}(M \mid\mid D)$.
4. Return signature $S \leftarrow (M_{\mathsf{MD}})^{d_{\mathsf{MD}}} \bmod N$.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Verify}(S, M, D, \mathsf{pk} \leftarrow N)$:

1. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
2. Compute $X \leftarrow S^{e_{\mathsf{MD}}} \bmod N$.
3. Return $\mathsf{Verify}_{\mathcal{M}}(M \mid\mid D, X)$.

Correctness. First, we show our above protocols are well-defined. In particular, we need to show that $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$ is actually computable. We will rely on Lemma 1 stating that all outputs of H_{MD} are co-prime with $\phi(N)$. Therefore, e_{MD} has a valid inverse modulo $\phi(N)$ and the signing algorithm is always well-defined.

Next, we will show that the verification succeeds for well-formed signatures. Let S be a signature that is produced by following the Setup and Sign algorithms properly for input message M and public metadata D. Then, we know that the signature satisfies:

$$S = (H_{\mathcal{M}}(M \mid\mid D))^{1/H_{\mathsf{MD}}(D)} \bmod N.$$

Then, the verification algorithm will return the following result

$$\mathsf{Verify}_{\mathcal{M}}(M \mid\mid D, S^{\cdot H_{\mathsf{MD}}(D)}) = \mathsf{Verify}_{\mathcal{M}}(M \mid\mid D, H_{\mathcal{M}}(M \mid\mid D)) = 1$$

as we know that $S^{H_{\mathsf{MD}}(D)} = H_{\mathcal{M}}(M \mid\mid D) \bmod N$ and $\mathsf{Verify}_{\mathcal{M}}(X, H_{\mathcal{M}}(X)) = 1$. Therefore, a well-formed signature will always be verified correctly.

5.3 Unforgeability

In this section, we prove the concrete security of our RSA signature with public metadata from the RSA assumption. In particular, we aim to answer the question of whether our new public metadata variant may have weaker security compared to standard RSA signatures. We essentially show that this is not the case as we show that the concrete security bounds are nearly identical to those proven by Bellare and Rogaway [13] with FDH message encodings except for an additive factor exponentially small in the security parameter λ.

At a high level, our security proof will construct an adversary $\mathcal{A}$ for the multi-exponent RSA game given exponents from $[1, 2^{\lambda-3}]$. Afterwards, we can apply Theorem 2 to obtain security based on the standard RSA assumption. However, we require that $e_{\mathbf{max}} \le 2^{\lambda/2}$ to ensure that the additive factor is exponentially small in λ. We start with the first step reducing security to the multi-exponent RSA assumption:

Lemma 2. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the $(\epsilon, t, q_{\mathsf{MD}})$-multi-exponent RSA assumption (Definition 8) with exponents in $[1, 2^{\lambda-3}]$ and the random oracle model, then $\mathsf{RSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2) where*

- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\epsilon_F = q_{\mathcal{M}} \cdot \epsilon$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. Let $\mathcal{A}_F$ be an adversary that can break the $(\epsilon_F, t_F, \ell_F)$-$\mathsf{SOMUF}$ of $\mathsf{RSA}_{\mathsf{MD}}$ with at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle, $H_{\mathcal{M}}$ and H_{MD}. That is, $\mathcal{A}_F$ can forge a signature with success probability of ϵ_F with running time of t_F. To complete the proof, we show that one can use $\mathcal{A}_F$ to construct an adversary $\mathcal{A}$ that can break the multi-exponent RSA game with exponents from $[1, 2^{\lambda-3}]$ using $O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$ additional running time and wins the game with probability at least $\epsilon_F/q_{\mathcal{M}}$.

For any adversary $\mathcal{A}$ trying to solve the multi-exponent RSA problem, $\mathcal{A}$ receives as input the challenge RSA exponents $(e_1, \ldots, e_{q_{\mathsf{MD}}})$, strong RSA modulus N and a challenge target X that is chosen uniformly at random $\mathbb{Z}_N^*$. Recall that the goal is to pick any $i \in [q_{\mathsf{MD}}]$ and output the e_i-th root of X modulo N. In other words, compute Y such that $Y^{e_i} = X \bmod N$.

$\mathcal{A}$ will execute the adversary $\mathcal{A}_F$ that outputs a forgery of the form $D, (S_i, M_i)_{i\in[x]}$. Without loss of generality, we will assume the following. First, $x \le \ell_F + 1$. As $\mathcal{A}_F$ makes at most ℓ_F signing oracle queries, $\mathcal{A}$ will only need to produce at most $\ell_F + 1$ valid signatures to win the strong one-more unforgeability game. Secondly, we will assume that $\mathcal{A}_F$ will have queried $H_{\mathcal{M}}(M_i)$ on all output messages. Again, this is without loss of generality as it will only increase the running time of $\mathcal{A}_F$ by at most $\ell_F + 1$ hash function queries. Furthermore, we will suppose that $\mathcal{A}_F$ will have queried both $H_{\mathcal{M}}(M)$ and $H_{\mathsf{MD}}(D)$ before submitting a signing query for M and D. Again, this is without loss of generality, as it only increases the running time of $\mathcal{A}_F$ by at most $2\ell_F$ hash queries.

During the execution of $\mathcal{A}_F$, $\mathcal{A}$ must successfully simulate queries for signing as well as the message encoding hash function $H_{\mathcal{M}}$ and the public metadata hash function H_{MD}. Before doing so, $\mathcal{A}$ keeps track of a count of the number of unique inputs to the message oracle $H_{\mathcal{M}}$. We know that at most $q_{\mathcal{M}}$ queries are made to $H_{\mathcal{M}}$. $\mathcal{A}$ will pick a random integer $z \leftarrow_R [q_{\mathcal{M}}]$ and will embed the challenge X as the output of the z-th query to $H_{\mathcal{M}}$. To keep track, $\mathcal{A}$ keeps counter $\mathsf{cnt}_{\mathcal{M}}$ to count the number of unique inputs to $H_{\mathcal{M}}$ where $\mathsf{cnt}_{\mathcal{M}}$ is initialized to zero. Similarly, $\mathcal{A}$ uses $\mathsf{cnt}_{\mathsf{MD}}$ to count the number of unique inputs to H_{MD} with $\mathsf{cnt}_{\mathsf{MD}}$ also initialized to zero. Finally, $\mathcal{A}$ keeps a map of inputs to important information for the hash function H_{MD} and $H_{\mathcal{M}}$ denoted by $\mathbf{M}_{\mathsf{MD}}$ and $\mathbf{M}_{\mathcal{M}}$. $\mathcal{A}$ will simulate each of these queries as follows.

For any query to $H_{\mathsf{MD}}(D)$:

1. If $\mathbf{M}_{\mathsf{MD}}[D]$ is set, return $\mathbf{M}_{\mathsf{MD}}[D]$.
2. Increment $\mathsf{cnt}_{\mathsf{MD}} \leftarrow \mathsf{cnt}_{\mathsf{MD}} + 1$.
3. Set $\mathbf{M}_{\mathsf{MD}}[D] \leftarrow e_{\mathsf{cnt}_{\mathsf{MD}}}$.
4. Return $\mathbf{M}_{\mathsf{MD}}[D]$.

For any query to $H_{\mathcal{M}}(M, D)$:

1. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
2. If $\mathbf{M}_{\mathcal{M}}[M, D]$ is set, return first value of tuple $\mathbf{M}_{\mathcal{M}}[M, D]$.
3. Increment $\mathsf{cnt}_{\mathcal{M}} \leftarrow \mathsf{cnt}_{\mathcal{M}} + 1$.
4. If $\mathsf{cnt}_{\mathcal{M}} = z$:
   (a) Set $\mathbf{M}_{\mathcal{M}}[M, D] \leftarrow (X, \perp)$.
5. Otherwise when $\mathsf{cnt}_{\mathcal{M}} \neq z$:
   (a) Generate random $A \leftarrow_R \mathbb{Z}_N^*$.
   (b) Compute $B = A^{e_{\mathsf{MD}}} \bmod N$.
   (c) Set $\mathbf{M}_{\mathcal{M}}[M, D] \leftarrow (B, A)$.

6. Return first value of tuple $\mathbf{M}_{\mathcal{M}}[M, D]$.

For any query to $\mathcal{O}^{\mathsf{Sign}}(M, D)$:

1. Retrieve $(B, A) \leftarrow \mathbf{M}_{\mathcal{M}}[M, D]$.
2. If $A = \perp$, abort.
3. Return A.

Finally, we will suppose that $\mathcal{A}$ does not abort and $\mathcal{A}_F$ outputs some forgery $D, (S_i, M_i)_{i \in [x]}$ while making strictly less than x signing queries for metadata D. If $H_{\mathcal{M}}(M_i \mid\mid D) = X$ for any $i \in [x]$, then we know the following holds:

$$H_{\mathcal{M}}(M_i \mid\mid D) = X = S_i^{H_{\mathsf{MD}}(D)}$$

assuming that $\mathcal{A}_F$ is successful at forging. Re-writing the above, we get that

$$S_i = X^{1/H_{\mathsf{MD}}(D)} = X^{1/e_j}$$

for some $j \in [q_{\mathsf{MD}}]$ and e_j is the j-th challenge exponent. In other words, this is a successfully decryption of the ciphertext X that would win the multi-exponent RSA game. Therefore, $\mathcal{A}$ will simply return (j, S_i) that would win the multi-exponent RSA game.

Adversarial Advantage. First, we show that the simulated view and real view of $\mathcal{A}_F$ are identical. In the real game, $\mathcal{A}_F$ receives random exponents from $[1, 2^{\lambda-3}]$. In the simulated game, $\mathcal{A}_F$ receives random challenge exponents $e_1, \ldots, e_{q_{\mathsf{MD}}}$ that are also random exponents from $[1, 2^{\lambda-3}]$. The view from $H_{\mathcal{M}}$ and the signing oracle are also identical as long as $\mathcal{A}$ does not abort.

To upper bound the aborting probability, we first note that $\mathcal{A}_F$ must return at least one message M_i such that (M_i, D) was never sent to the signing oracle. Assuming that $\mathcal{A}$ successfully executed $\mathcal{A}_F$ without aborting, the probability that public metadata D and message M_i are output by $\mathcal{A}_F$ where $H_{\mathcal{M}}(M_i \mid\mid D) = X$ such that (M_i, D) was not a signing oracle query by $\mathcal{A}_F$ is at least $1/q_{\mathcal{M}}$. This is because $\mathcal{A}_F$ makes at most $q_{\mathcal{M}}$ queries to $H_{\mathcal{M}}$ and must output at least one message. In this case, it is clear that $\mathcal{A}$ will not abort when executing $\mathcal{A}_F$. Finally, we note that $\mathcal{A}$ produces a forgery with probability at most ϵ_F meaning that $\mathcal{A}$ wins the RSA game with probability at least $\epsilon_F / q_{\mathcal{M}}$.

Adversarial Running Time. Suppose that $\mathcal{A}_F$ runs in time t_F. By our assumptions, we note that we increase the running time of $\mathcal{A}_F$ by at most $O(\ell_F)$ hash queries. For each signing query, $\mathcal{A}$ must perform a single exponentiation. For each query to $H_{\mathcal{M}}$ and H_{MD}, $\mathcal{A}$ requires $O(1)$ time to simulate each answer correctly. Therefore, $\mathcal{A}$ requires $t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$ time. □

Using the above lemma, we can apply Theorem 2 to reduce security to standard RSA.

Theorem 4. *Suppose that $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ correspond to full domain hash (FDH) message encoding. Assuming the (ϵ, t)-RSA assumption (Definition 6) with exponents in $[1, e_{\mathbf{max}}]$ with $e_{\mathbf{max}} \leq 2^{\lambda/2}$ and the random oracle model, then $\mathsf{RSA}_{\mathsf{MD}}$ is $(\epsilon_F, t_F, \ell_F)$-strong one-more unforgeable (Definition 2) where*

- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\epsilon_F = q_{\mathcal{M}} \cdot \epsilon + \frac{q_{\mathcal{M}} \cdot q_{\mathsf{MD}} \cdot e_{\mathbf{max}}}{2^{\lambda-3}} \leq q_{\mathcal{M}} \cdot \epsilon + 2^{-\lambda/2 + \log q_{\mathcal{M}} + \log q_{\mathsf{MD}} + 3}$

and the adversary $\mathcal{A}$ runs in time at most t_F and makes at most ℓ_F, $q_{\mathcal{M}}$ and q_{MD} queries to the signing oracle and hash functions $H_{\mathcal{M}}$ and H_{MD} respectively.

Proof. First, we apply Lemma 2 to reduce security to the $(\epsilon_F/q_{\mathcal{M}}, t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F), q_{\mathsf{MD}})$-multi-exponent RSA assumption with exponents from $[1, 2^{\lambda-3}]$. By applying Theorem 2, we obtain that this is equivalent to the $(\epsilon_F/q_{\mathcal{M}} - q_{\mathsf{MD}} \cdot (e_{\mathbf{max}}/2^{\lambda-3}), t_F + O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F))$-RSA assumption with exponents from $[1, e_{\mathbf{max}}]$. By plugging in the appropriate $e_{\mathbf{max}} \leq 2^{\lambda/2}$, we can complete the proof. □

Smaller Choices of $e_{\mathbf{max}}$. The above theorem makes the very loose restriction that $e_{\mathbf{max}} \leq 2^{\lambda/2}$. In practice, $e_{\mathbf{max}}$ is typically much smaller. For example, implementations of RSA typically use $e = 3$ or $e = 65537$. In this case, we can set $e_{\mathbf{max}} = 3$ or $e_{\mathbf{max}} = 65537$ to get much tighter security bounds. For $e_{\mathbf{max}} = 3 < 2^2$, we can get the bound on $\epsilon_F = t_F \cdot \epsilon + 2^{-\lambda + \log q_{\mathcal{M}} + \log q_{\mathsf{MD}} + 5}$. Similarly, for $e_{\mathbf{max}} = 65537 = 2^{16} + 1 < 2^{17}$, we can get $\epsilon_F = t_F \cdot \epsilon + 2^{-\lambda + \log q_{\mathcal{M}} + \log q_{\mathsf{MD}} + 20}$.

Comparison with Standard RSA Signatures. We note that the concrete security of our variant with public metadata has almost identical bounds as those proven in [13] for RSA signatures. Re-formulating their analysis into our terminology, they showed that RSA signatures with FDH encoding has $\epsilon_F = q_{\mathcal{M}} \cdot \epsilon$ assuming the (ϵ, t)-RSA assumption with at most $q_{\mathcal{M}}$ queries to $H_{\mathcal{M}}$. For our version with public metadata, we lose an additive factor of $2^{-\lambda/2 + \log q_{\mathcal{M}} + \log q_{\mathsf{MD}} + 3}$ when assuming $e_{\mathbf{max}} \leq 2^{\lambda/2}$. In other words, this provides evidence that our public metadata variant of RSA signatures provides similar unforgeability guarantees as standard RSA signatures.

Encoding of Message and Public Metadata. We note that our proof critically uses the fact that we pass both the message and public metadata into the FDH encoding algorithm. When $\mathcal{A}$ programs the random oracle for $H_{\mathcal{M}}$, $\mathcal{A}$ exponentiates according to $H_{\mathsf{MD}}(D)$. This is only possible because D is also passed as an input to $H_{\mathcal{M}}$. If, instead, we only passed the message M into $H_{\mathcal{M}}$, then the best that $\mathcal{A}$ can do is simply aim to guess the public metadata that will be later sent with M to the signing oracle. As there is no requirement that $\mathcal{A}_F$ has even picked D yet, it is impossible for $\mathcal{A}$ to guess this correctly. Therefore, it is integral that our algorithm passes the public metadata D into $H_{\mathcal{M}}$ for security.

Extension to PSS Encoding. Finally, we note that our proof can be modified when using PSS message encoding. In particular, the random oracle $H_{\mathcal{M}}$ is currently programmed in a trivial way to output random elements that is indistinguishable from FDH encodings. Instead, we can follow the identical proof techniques of Bellare and Rogaway [13] to program $H_{\mathcal{M}}$ to follow the PSS encoding. The main difference is the following. In the FDH proof, the adversary had to pick one of the query to $H_{\mathcal{M}}$ to choose to embed the random input target. By using the structure of PSS encoding, the adversary can instead embed the random input target into every query to $H_{\mathcal{M}}$. This significantly improves the ability of the adversary to win the game. As a result, we can obtain identical concrete security when using PSS encoding as those proved in [13] except for a similar additive factor exponentially small in λ when simulating the public metadata hash function H_{MD}.

6 RSA Blind Signatures with Public Metadata

In this section, we present our main protocol for RSA blind signatures with public metadata $\mathsf{BlindRSA}_{\mathsf{MD}}$. We also prove the unlinkability and unforgeability of our scheme.

6.1 Protocol

We will present our RSA blind signature with public metadata formally. We will use the same public metadata hash function H_{MD} from Section 5.2. Additionally, we will assume some message encoding protocol defined by $H_{\mathcal{M}}$ and $\mathsf{Verify}_{\mathcal{M}}$. We point readers back to Section 5.1 for more details on various options such as the FDH and PSS algorithms. As a note, we will utilize a modification presented by Lysyanskaya [25] that enables providing unlinkability even against malicious signers. In particular, it was shown that appending a random string to the message of bit length κ is sufficient to provide anonymity against even adversarially chosen RSA modulus. We incorporate this technique into our protocol as well.

$\mathsf{RSA}_{\mathsf{MD}}.\mathsf{Setup}(1^\lambda)$:

1. Randomly generate $(N, p, q) \leftarrow_R \mathsf{Gen}(1^\lambda)$.
2. Compute $\phi(N) \leftarrow (p-1)(q-1)$.
3. Return $\mathsf{pk} \leftarrow N, \mathsf{sk} \leftarrow \phi(N)$.

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Blind}(M, D, \mathsf{pk} \leftarrow N)$:

1. Pick a random string rand of length κ.
2. Set $M' \leftarrow M \mid\mid \mathsf{rand}$.
3. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
4. Pick R uniformly at random from $\mathbb{Z}_N^*$.
5. Set state $\mathsf{st} \leftarrow (\mathsf{rand}, R)$.
6. Compute *blinded message* $B_M \leftarrow R^{e_{\mathsf{MD}}} \cdot H_{\mathcal{M}}(M' \mid\mid D) \bmod N$.
7. Return (st, B_M).

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Sign}(B_M, D, \mathsf{pk} \leftarrow N, \mathsf{sk} \leftarrow \phi(N))$:

1. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
2. Compute $d_{\mathsf{MD}} \leftarrow (e_{\mathsf{MD}})^{-1} \bmod \phi(N)$.
3. Return $S' \leftarrow (B_M)^{d_{\mathsf{MD}}} \bmod N$.

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Finalize}(\mathsf{st} \leftarrow (\mathsf{rand}, R), S', M, D, \mathsf{pk} \leftarrow N)$:

1. Compute $S \leftarrow S' \cdot R^{-1} \bmod N$.
2. If $\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Verify}((S, \mathsf{rand}), M, D, \mathsf{pk}) = 0$, return $\perp$.[5]
3. Return *signature* (S, rand).

$\mathsf{BlindRSA}_{\mathsf{MD}}.\mathsf{Verify}((S, \mathsf{rand}), M, D, \mathsf{pk} \leftarrow N)$:

1. Compute $M' \leftarrow M \mid\mid \mathsf{rand}$.
2. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
3. Compute $X \leftarrow S^{e_{\mathsf{MD}}} \bmod N$.
4. Return $\mathsf{Verify}_{\mathcal{M}}(M' \mid\mid D, X)$.

Correctness. By Lemma 1, we know that the algorithm above is well-defined as the signer can always compute an inverse of $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. To show that the above protocol correctly verifies well-formed signatures, we can consider an execution of the blind signing protocol. The blinded message output by Blind will be

$$R^{e_{\mathsf{MD}}} \cdot H_{\mathcal{M}}(M' \mid\mid D).$$

The output of Sign will be

$$(R^{e_{\mathsf{MD}}} \cdot H_{\mathcal{M}}(M' \mid\mid D))^{(e_{\mathsf{MD}})^{-1}} = R \cdot H_{\mathcal{M}}(M' \mid\mid D)^{(e_{\mathsf{MD}})^{-1}}.$$

Finally, the output of $\mathsf{Finalize}$ is

$$H_{\mathcal{M}}(M' \mid\mid D)^{(e_{\mathsf{MD}})^{-1}}.$$

Consider a final execution of Verify that will output

$$\mathsf{Verify}_{\mathcal{M}}(M' \mid\mid D, H_{\mathcal{M}}(M' \mid\mid D)^{(e_{\mathsf{MD}})^{-1} \cdot e_{\mathsf{MD}}}) = \mathsf{Verify}_{\mathcal{M}}(M' \mid\mid D, H_{\mathcal{M}}(M' \mid\mid D)) = 1.$$

Therefore, a well-formed signature will always be verified correctly.

[5]This is optional for correctness but should be done to check that the server signed with the correct key.

6.2 Unforgeability

Next, we prove the unforgeability of $\mathsf{BlindRSA}_{\mathsf{MD}}$. To do this, we will show that any adversary that can forge signatures in $\mathsf{BlindRSA}_{\mathsf{MD}}$ is able to break the chosen-target, restricted-exponent RSA inversion assumption. At a high level, we show that one can simulate signing oracle queries using the RSA decryption oracle in the chosen-target, restricted-exponent RSA inversion game.

Theorem 5. *Suppose that* $(H_{\mathcal{M}}, \mathsf{Verify}_{\mathcal{M}})$ *correspond to full domain hash (FDH) message encoding. Assuming the* (ϵ, t, ℓ)*-chosen-target, restricted-exponent RSA inversion assumption (Definition 9) with exponents in* $[1, e_{\mathbf{max}}]$ *such that* $e_{\mathbf{max}} = 2^{\lambda-3}$ *and the random oracle model, then* $\mathsf{BlindRSA}_{\mathsf{MD}}$ *is* $(\epsilon_F, t_F, \ell_F)$*-strong one-more unforgeable (Definition 2) where*

- $\epsilon_F = \epsilon + 2^{-\lambda+2\log q_{\mathcal{M}}}$
- $t_F = t - O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$
- $\ell_F = \ell$

and the adversary $\mathcal{A}$ *runs in time at most* t_F *and makes at most* ℓ_F*,* $q_{\mathcal{M}}$ *and* q_{MD} *queries to the signing oracle and hash functions* $H_{\mathcal{M}}$ *and* H_{MD} *respectively.*

Proof. To prove this, we will show that if there exists some adversary $\mathcal{A}_F$ that successfully forges signatures for $\mathsf{BlindRSA}_{\mathsf{MD}}$ to win the strong one-more unforgeability game, then we can use $\mathcal{A}_F$ to construct an adversary $\mathcal{A}$ to break the chosen-target, restricted-exponent RSA inversion assumption. The forger $\mathcal{A}_F$ runs in time t_F, wins the game with probability ϵ_F and uses at most ℓ_F signing queries. Our reduction will only increase the running time of the adversary $\mathcal{A}$ by a factor of $O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$, reduces the winning probability by at most $2^{-\lambda+2\log q_{\mathcal{M}}}$ and maintains the same number of oracle queries between signing and RSA decryption.

Our adversary $\mathcal{A}$ will simulate $\mathcal{A}_F$ to break the chosen-target, restricted-exponent RSA game. Note that $\mathcal{A}$ needs to be able to successfully simulate all hash and signing oracle queries performed by $\mathcal{A}_F$. $\mathcal{A}$ simulates these queries as follows:

For any query to $H_{\mathsf{MD}}(D)$:

1. If $\mathbf{M}_{\mathsf{MD}}[D]$ is set, return $\mathbf{M}_{\mathsf{MD}}[D]$.
2. Compute $\mathbf{M}_{\mathsf{MD}}[D] \leftarrow \mathcal{O}^{\mathsf{exp}}()$.
3. Return $\mathbf{M}_{\mathsf{MD}}[D]$.

For any query to $H_{\mathcal{M}}(M, D)$:

1. If $\mathbf{M}_{\mathcal{M}}[M, D]$ is set, return $\mathbf{M}_{\mathcal{M}}[M, D]$.
2. Compute $\mathbf{M}_{\mathcal{M}}[M, D] \leftarrow \mathcal{O}^{\mathcal{X}}()$.
3. Return $\mathbf{M}_{\mathcal{M}}[M, D]$.

For any query to $\mathcal{O}^{\mathsf{Sign}}(X, D)$:

1. Compute $e_{\mathsf{MD}} \leftarrow H_{\mathsf{MD}}(D)$.
2. Return $\mathcal{O}^{\mathsf{RSA}}(X, e_{\mathsf{MD}})$.

Adversarial Advantage. First, we note that the simulated outputs of $\mathcal{O}^{\mathsf{exp}}$ and the real outputs of H_{MD} are identical as $e_{\mathbf{max}} = 2^{\lambda-3}$. Similarly, $\mathcal{O}^{\mathcal{X}}$ returns random elements identical to the FDH encoding. Finally, we note that signing always returns a valid signature as $\mathcal{O}^{\mathsf{RSA}}$ only receives exponents that are output by $\mathcal{O}^{\mathsf{exp}}$.

Finally, $\mathcal{A}_F$ will output the values $D, (S_i, M_i)_{i\in[x]}$ where $x \geq \mathsf{cnt}_D + 1$ such that cnt_D is the maximum number of signing oracles performed with public metadata D. If $\mathcal{A}_F$ successfully forges, then we know that $S_i^{H_{\mathsf{MD}}(D)} = H_{\mathcal{M}}(M_i \mid\mid D)$ for all $i \in [x]$. Then, $\mathcal{A}$ will output $H_{\mathsf{MD}}(D), (S_i, H_{\mathcal{M}}(M_i \mid\mid D))_{i\in[x]}$ as its output to the chosen-target, restricted-exponent RSA inversion game. Note, $\mathcal{A}$ wins the game as long as they are all valid decryptions and there are no collisions in the output of $H_{\mathcal{M}}(M_i \mid\mid D)$. As there are at most $q_{\mathcal{M}}$ queries to $H_{\mathcal{M}}$, we know the collision probability is at most $q_{\mathcal{M}}^2/2^{-\lambda} = 2^{-\lambda+2\log q_{\mathcal{M}}}$ since $|\mathbb{Z}_N^*| \geq 2^{-\lambda}$.

Therefore, we know that $\mathcal{A}$ wins the game if $\mathcal{A}_F$ wins the unforgeability game and there are no collisions in the outputs of $H_{\mathcal{M}}$. So, the probability that $\mathcal{A}$ wins is $\epsilon_F - 2^{-\lambda+2\log q_{\mathcal{M}}}$.

Running Time. Next, we analyze the running time of $\mathcal{A}$. We know that $\mathcal{A}_F$ runs time at most t_F. For each hash and signing oracle query, $\mathcal{A}$ performs $O(1)$ additional operations to correctly simulate responses. As there are most $q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F$ queries overall, we see that $\mathcal{A}$ requires $O(q_{\mathcal{M}} + q_{\mathsf{MD}} + \ell_F)$ additional time. Furthermore, we note that each signing query results in exactly one decryption oracle query. Therefore, we know that at most ℓ_F decryption oracle queries are made. □

Message Encoding Discussion. In Section 5.3, we mentioned that using the PSS message encoding as opposed to FDH message encoding would improve concrete security (see Section 5.1 for details on message encoding). To our knowledge, there is no benefit to concrete security when plugging in PSS for the security proof (Theorem 5) of our RSA blind signatures with public metadata. It is possible to program $H_{\mathcal{M}}$ such that its output will simulate PSS encodings (using the techniques from [12]), but the resulting reduction would have identical parameters. This phenomenon also exists in standard RSA blind signatures as the original work [12] proved security using only FDH as PSS did not provide any security improvements.

6.3 Unlinkability

To prove that $\mathsf{BlindRSA}_{\mathsf{MD}}$ provides unlinkability, we will rely on prior results in [25] that show that appending random strings to the message is sufficient to provide protections against maliciously generated parameters. At a high level, the work in [25] shows that for any maliciously chosen RSA modulus N and public exponent e, unlinkability for standard RSA blind signatures with appended random strings of bit length κ holds with probability except $t \cdot 2^{-\kappa}$ for adversaries running in time t.

For our case, the adversary is able to choose the public metadata D. However, this ends up being a more restrictive way for the adversary to choose as the final exponent $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. Clearly, it is an easier setting for the adversary to choose e_{MD} directly as opposed to D. As a result, we can rely on the results from [25] almost immediately as follows.

Theorem 6. *Assuming the random oracle model,* $\mathsf{BlindRSA}_{\mathsf{MD}}$ *satisfies* $(t{\cdot}2^{-\kappa}, t)$*-unlinkability (Definition 3).*

Proof. Consider the unlinkability games for $\mathsf{BlindRSA}_{\mathsf{MD}}$ with public metadata and the standard RSA blind signatures in [25]. We note there is only one difference between the two settings. In the former, the adversary must choose D. The final RSA public exponent becomes $e_{\mathsf{MD}} = H_{\mathsf{MD}}(D)$. In the latter, the adversary is able to directly choose the final public exponent. Note, we can do this because $\mathsf{BlindRSA}_{\mathsf{MD}}$ is nearly identical to standard RSA blind signatures after augmented the public exponent to be $H_{\mathsf{MD}}(D)$. Clearly, the scenarios studied in [25] is more favorable for the adversary.

Suppose there exists an adversary $\mathcal{A}$ that can compromise the unlinkability of $\mathsf{BlindRSA}_{\mathsf{MD}}$ with probability ϵ and running time t. Note, $\mathcal{A}$ outputs values public key N, messages M_0 and M_1 and public metadata D in the first step. We construct $\mathcal{A}'$ that instead outputs $e = H_{\mathsf{MD}}(D)$ as the public exponent while keeping the RSA modulus N and messages M_0, M_1 identical. As the remainder of the games are identical, $\mathcal{A}'$ also wins with probability ϵ and running time t. □

Modulus Bit Length	**Algorithm**	**Public Metadata**	**No Public Metadata**
2048	Blind	1.10	0.03
2048	Sign	3.19	1.06
2048	Finalize	1.08	0.02
2048	Verify	1.08	0.02
3072	Blind	3.48	0.05
3072	Sign	10.19	2.73
3072	Finalize	3.47	0.05
3072	Verify	3.45	0.05
4096	Blind	8.14	0.09
4096	Sign	23.70	6.18
4096	Finalize	7.98	0.08
4096	Verify	7.92	0.09

Figure 10: A comparison table of RSA blind signature protocols with and without public metadata. All computational costs are presented in milliseconds (ms).

7 Experimental Evaluation

In this section, we present an experimental evaluation of our RSA blind signatures with public metadata. In particular, we show that our protocol has similar performance to RSA blind signatures without public metadata augmentation.

Our Implementation. Our open-source code can be found at [1] containing an implementation of RSA blind signatures with and without public metadata. Both implementations utilize PSS message encodings and follow their respective specifications [10, 26] for other details such as instantiating hash functions and ensuring domain separation. Our implementations rely on BoringSSL for many underlying cryptographic primitives such as basic RSA functionality.

Experimental Setup. We conducted our experiments using Ubuntu PCs with 12 cores, 3.7 GHz Intel Xeon W-2135 and 64 GB of RAM. Our experiments are all executed using a single thread. All reported results are the average of at least 10 experimental trials with standard deviation less than 10% of the averages. In our experiments, we instantiate both implementations with RSA modulus of bit-length 2048 and 4096. The appended random string to the message for unlinkabiity will be $\kappa = 256$ bits. Finally, we use SHA384 as the underlying hash function with 384-bit salts.

Comparison. We report all our results of the computational costs for the sub-routines Blind, Sign, Finalize and Verify in Figure 10. In general, we note that the protocol with public metadata is slower. We attribute this to two reasons. First, Sign requires an additional inverse operation modulo $\phi(N)$ to compute the private exponent. The other main reason is that the RSA functionalities in BoringSSL are optimized for small RSA exponents. For the public metadata protocol, the output of H_{MD} can be quite large that is needed for security. Therefore, it is not surprising that the public metadata variant is slower. Nevertheless, the RSA blind signatures with public metadata are still more than sufficiently fast enough for real-world applications. For example, the public metadata protocol with a 2048-bit modulus requires approximately 1 millisecond for Blind, Finalize and Verify and approximately 3 milliseconds for Sign.

For communication, we note that both protocols have identical signature sizes that only depends on the length of the modulus. Therefore, we omit presenting these results.

8 Conclusion and Open Problems

In this paper, we present a new protocol for RSA blind signatures that enables augmentation with public metadata. We provide strong evidence that the concrete security of our scheme is nearly identical to standard

RSA blind signatures and prove security of our schemes. Furthermore, we show that our public metadata protocol is concretely efficient with comparable overhead as standard RSA blind signatures. We leave the following open questions:

- Can one prove security of our variant of RSA blind signatures with public metadata from the chosen-target RSA inversion assumption (Definition 7) instead of the chosen-target, restricted-exponent RSA inversion assumption (Definition 9) that we introduced? This seems possible since we are able to prove the non-blind version of RSA signatures with public metadata directly from the RSA assumption.

- What is the relationship between the chosen-target (Definition 7) and chosen-target, restricted-exponent (Definition 9) variants of the RSA inversion assumption?

- What is the relationship between the chosen-target, restricted-exponent (Definition 9) and chosen-target, chosen-exponent (Definition 11) variants of the RSA inversion assumption?

Acknowledgements. The authors would like to thank Scott Hendrickson for proposing the problem, Cathie Yun for help implementing standard RSA blind signatures, and David Benjamin, Steven Valdez and Chris Wood for helping review the protocol and our implementations. The authors would also like to thank Sarvar Patel and Giuseppe Persiano for very insightful discussions on the new RSA assumptions and feedback on earlier versions of this paper.

References

[1] Anonymous tokens (AT). https://github.com/google/anonymous-tokens.

[2] BoringSSL. https://github.com/google/boringssl.

[3] Getting started with trust tokens. https://web.dev/trust-tokens/.

[4] iCloud private relay overview. https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf.

[5] Introducing private click measurement, PCM. https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/.

[6] VPN by Google One, explained. https://one.google.com/about/vpn/howitworks.

[7] Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 262–279. Springer, Heidelberg, February 2006.

[8] Masayuki Abe and Jan Camenisch. Partially blind signature schemes. In *1997 Symposium on Cryptography and Information Security*, 1997.

[9] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, Heidelberg, November 1996.

[10] Ghous Amjad, Scott Hendrickson, Christopher Wood, and Kevin Yeo. Partially blind RSA signatures. https://datatracker.ietf.org/doc/draft-amjad-cfrg-partially-blind-rsa/, 2023.

[11] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, Heidelberg, May 1997.

[12] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In Paul F. Syverson, editor, *FC 2001*, volume 2339 of *LNCS*, pages 319–338. Springer, Heidelberg, February 2002.

[13] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.

[14] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[15] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[16] David Chaum. Blind signature system. In David Chaum, editor, *CRYPTO'83*, page 153. Plenum Press, New York, USA, 1983.

[17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 321–350. Springer, Heidelberg, April / May 2017.

[18] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

[19] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

[20] Frank Denis, Frederic Jacobs, and Christopher Wood. RSA blind signatures. https://datatracker.ietf.org/doc/draft-irtf-cfrg-rsa-blind-signatures/, 2023.

[21] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Heidelberg, August 1997.

[22] Scott Hendrickson and Christopher Wood. Privacy pass: Public metadata issuance. https://datatracker.ietf.org/doc/draft-hendrickson-privacypass-public-metadata/, 2023.

[23] Sharon Huang, Subodh Iyengar, Sundar Jeyaraman, Shiv Kushwah, Chen-Kuei Lee, Zutian Luo, Payman Mohassel, Ananth Raghunathan, Shaahid Shaikh, Yen-Chieh Sung, et al. Dit: De-identified authenticated telemetry at scale, 2021.

[24] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 308–336. Springer, Heidelberg, August 2020.

[25] Anna Lysyanskaya. Security analysis of RSA-BSSA. In *IACR International Conference on Public-Key Cryptography*, pages 251–280. Springer, 2023.

[26] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. PKCS #1: RSA cryptography specifications version 2.2. https://datatracker.ietf.org/doc/html/rfc8017, 2016.

[27] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 252–265. Springer, Heidelberg, November 1996.

[28] Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 662–679. Springer, Heidelberg, May 2012.

[29] Atle Selberg. An elementary proof of the prime-number theorem. *Annals of Mathematics*, pages 305–313, 1949.

[30] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 207–220. Springer, 2000.

[31] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[32] Tjerand Silde and Martin Strand. Anonymous tokens with public metadata and applications to private contact tracing. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 179–199. Springer, 2022.

[33] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. Efficient verifiably encrypted signature and partially blind signature from bilinear pairings. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 191–204. Springer, Heidelberg, December 2003.

A Discussion about Strong RSA Modulus

In practice today, the Gen algorithm is used to generate only the RSA modulus N. The distribution $\mathcal{D}_N$ to generate the public exponent is typically a singleton element. To generate the RSA modulus N, the standard approach is to pick two random prime numbers of length λ that are sufficiently large. To be sufficiently large, the prime numbers must be at least $\lfloor 2^{\lambda-1/2} \rfloor$. For more details, we point to the RSA modulus generation algorithm within BoringSSL [2].

Generating Strong RSA Modulus. For our new protocol, we need to modify the Gen algorithm to always output strong RSA modulus $N = pq$ where p and q are safe primes. The straightforward approach is to use the same Gen algorithm of generating random primes and simply check whether the chosen primes are safe. For any prime p, one can simply using primality testing algorithms to check whether $(p-1)/2$ is prime to see if p is a safe prime. As safe primes are more scarce than primes, this algorithm will be slower. However, we can estimate the asymptotic additional time needed to generate safe primes compared to normal primes. By the prime number theorem [29], we know that the density of prime numbers is $O(1/\lambda)$ for numbers at most 2^λ. For safe primes, we can rely on one of several conjectures such as Dickson's conjecture or the twin prime conjecture (see [31]) that state the density of safe primes is $O(1/\lambda^2)$. Therefore, we could expect that the Gen algorithm finds a safe prime after generating $O(\lambda)$ random primes.

Comparing Assumptions. In Section 3.2, we present two RSA assumptions: the standard one and one restricted to only strong RSA modulus. We now prove the relation of these two assumptions that was formally presented in Theorem 1. At a high level, we will use the same ideas as the prior section comparing the densities of primes and safe primes.

Proof of Theorem 1. First, we will assume there is an adversary $\mathcal{A}$ that breaks the strong modulus version of the assumption with probability ϵ and running time t. We show that $\mathcal{A}$ will also break the standard RSA game with probability $O(\epsilon/\lambda^2)$ and running time t to prove the statement. Recall that λ is the length of the prime factors of the RSA modulus.

To prove this, we simply analyze the probability that a randomly generated RSA modulus is a strong RSA modulus. Assuming Dickson's conjecture [31], the probability that a randomly generated prime number of length λ is also a safe prime is $\Theta(1/\lambda)$. For a RSA modulus to be strong, both prime factors must be safe. Therefore, this occurs with probability $\Theta(1/\lambda^2)$. Conditioned on the RSA modulus being strong, the adversary $\mathcal{A}$ wins the standard RSA game with probability ϵ. Therefore, $\mathcal{A}$ wins the game with probability at least $O(\epsilon/\lambda^2)$. □

To our knowledge, there is no straightforward relation in the opposite direction. In particular, it is possible that the RSA assumption for strong modulus is true while the standard RSA assumption is false. This could be possible if there exists an attack that targets specifically non-strong RSA modulus. We leave it as an open question to determine the relation in this direction.

B Equivalence of Unforgeability Definitions

In this section, we consider the alternative definition of unforgeability with public metadata that was introduced in [32]. At a high level, the difference is that this alternative definition does not limit the total number of queries explicitly. Instead, the adversary is limited to at most q oracle signing queries for each choice of public metadata D. We present the security game in Figure 11 and the formal definition below:

Definition 13 ((ϵ, t, ℓ)-Alternative Strong One-More Unforgeableability with Public Metadata)**.** *Let λ be the security parameter and consider the game $\mathsf{G}^{\mathsf{ASOMUF}}_{\mathsf{Tok},\mathcal{A},\ell}$ in Figure 11. A token scheme* Tok *is (ϵ, t, ℓ)-alternative strong one-more unforgeable if, for any adversary $\mathcal{A}$ that runs in probabilistic time t and for any $\ell \geq 0$,*

$$\Pr[\mathsf{G}^{\mathsf{ASOMUF}}_{\mathsf{Tok},\mathcal{A},\ell}(\lambda) = 1] \leq \epsilon.$$

```
Game $\mathsf{G}^{\mathsf{ASOMUF}}_{\mathsf{Tok},\mathcal{A},\ell}(\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Tok.Setup}(1^\lambda)$
Initialize $\mathsf{cnt}_D \leftarrow 0$ for all choices of public metadata $D$.
$D, (S_i, M_i)_{i \in [\ell+1]} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{crs}, \mathsf{pk})$
Return 1 if and only if all the following hold:
  - $\forall i \neq j \in [\ell+1], S_i \neq S_j \vee M_i \neq M_j$
  - $\forall i \in [\ell+1], \mathsf{Tok.Verify}(S_i, M_i, D, \mathsf{pk}) = 1$
```

```
Oracle $\mathcal{O}^{\mathsf{Sign}}(\mathsf{msg}, D)$:
If $\mathsf{cnt}_D \geq \ell$:
  Return $\perp$.
$\mathsf{cnt}_D \leftarrow \mathsf{cnt}_D + 1$
Return $\mathsf{Tok.Sign}(\mathsf{msg}, D, \mathsf{pk}, \mathsf{sk})$.
```

Figure 11: Alternative Strong One-More Unforgeability (ASOMUF) Game with Public Metadata.

We show that the two definitions are essentially equivalent up to some loss in the number of oracle queries due to being unable to bound the total number of queries executed by the adversary in the alternative definition.

Theorem 7. *A token scheme* Tok *satisfies alternative strong one-more unforgeability (Definition 13) if and only if the token scheme* Tok *also satisfies strong ome-more unforgeability (Definition 2). The reductions only lose up to $O(t+\ell)$ additive factors in the adversary's running time and number of oracle signing queries.*

Proof. Assume Tok does not satisfy (ϵ, t, ℓ)-alternative strong one-more unforgeability and there exists an adversary $\mathcal{A}$ that wins the game $\mathsf{G}^{\mathsf{ASOMUF}}$. The same adversary also wins the game $\mathsf{G}^{\mathsf{SOMUF}}$ meaning Tok is not (ϵ, t, t)-strong one-more unforgeable as the maximum number of oracle queries sent by the adversary is at most its running time.

For the other direction, assume that Tok is not (ϵ, t, ℓ)-strong one-more unforgeable. Suppose adversary $\mathcal{A}$ wins the game $\mathsf{G}^{\mathsf{SOMUF}}$. If the output of $\mathcal{A}$ consists of $\ell + 1$ tuples, then it also wins $\mathsf{G}^{\mathsf{ASOMUF}}$. Otherwise, we can afford to use additional queries to the signing oracle in $\mathsf{G}^{\mathsf{ASOMUF}}$. Suppose that $\mathcal{A}$ outputs $\mathsf{cnt}_D + 1$ tuples using at most cnt_D signing oracle queries for public metadata D. Then, we can win $\mathsf{G}^{\mathsf{SOMUF}}$ by the adversary making an additional $\ell - \mathsf{cnt}_D - 1$ signing oracle queries for different messages. Therefore, Tok is not $(\epsilon, t + O(\ell), 2\ell)$-alternative strong one-more unforgeable as we bound the additional running time by $O(\ell)$ and the additional number of signing oracle queries by at most ℓ. □