The BiBa One-Time Signature and Broadcast Authentication Protocol

Adrian Perrig*
University of California, Berkeley
Digital Fountain
perrig@cs.berkeley.edu

ABSTRACT

We introduce the *BiBa signature* scheme, a new signature construction that uses one-way functions without trapdoors. BiBa features a low verification overhead and a relatively small signature size. In comparison to other one-way function based signature schemes, BiBa has smaller signatures and is at least twice as fast to verify (which probably makes it one of the fastest signature scheme to date for verification). On the downside, the BiBa public key is large, and the signature generation overhead is higher than previous schemes based on one-way functions without trapdoors (although it can be trivially parallelized).

One of the main challenges of securing broadcast communication is source authentication, which allows all receivers to verify the origin of the data. An ideal broadcast authentication protocol should be efficient for the sender and the receiver, have a small communication overhead, allow the receiver to authenticate each individual packet, provide perfect robustness to packet loss, scale to large numbers of receivers, and provide instant authentication (no buffering of data at the sender or receiver side). We are not aware of any previous protocol that satisfies all these properties. We present the BiBa broadcast authentication protocol, a new construction based on the BiBa signature, that achieves all our desired properties, with the tradeoff that it requires a moderate computation overhead for the sender to generate the authentication information, and that it requires loose time synchronization between the sender and receivers.

Keywords: Broadcast authentication, source authentication for multicast, one-time signature, signature based on a one-way function without trapdoor.

*We gratefully acknowledge funding support for this research. This research was sponsored in part the United States Postal Service (contract USPS 102592-01-Z-0236), by the United States Defense Advanced Research Projects Agency (contract N66001-99-2-8913), by the United States National Science Foundation (grant FD99-79852), and by Digital Fountain. DARPA Contract N66001-99-2-8913 is under the supervision of the Space and Naval Warfare Systems Center, San Diego. This paper represents the opinions of the authors and does not necessarily represent the opinions or policies, either expressed or implied, of the United States government, of DARPA, NSF, USPS, any of its agencies, or of Digital Fountain.

CCS'01, November 5-8, 2001, Philadelphia, Pennsylvania, USA.

1. INTRODUCTION

For the past 25 years researchers have created and refined digital signature schemes using one-way functions without trapdoors [2, 5, 9, 10, 13, 14, 18, 23]. These signature schemes are efficient for signature generation and verification, but the signatures are too large for many applications. We propose the *BiBa signature*, a new approach for signatures based on one-way functions without trapdoor. The signature size of our scheme is much smaller than most previous signatures based on one-way functions; and the verification is also more efficient. However, our public keys are larger than most previous systems, and the time to generate signatures is also higher.

Our new signature scheme immediately yields important new applications. In particular, we extend the BiBa signature scheme to design a new protocol for authenticating broadcasts, such as streaming information broadcast over the Internet. Many applications need to authenticate broadcast data, i.e. verify the data origin. The main challenges to design an efficient broadcast authentication protocol are:

- *Efficient generation and verification.* The generation and verification overhead for the authentication information should be small. It is important that the verification overhead is small, since a large number of receivers need to verify the authentication information, and some receivers might have restricted computation power.
- *Real-time/instant authentication.* Many applications such as stock quote broadcasts require real-time data authentication. Hence, neither the sender nor the receiver should buffer data messages before sending or verification.
- *Individual message authentication.* The receiver can authenticate each individual message it receives.
- *Robustness to packet loss.* Internet broadcasts can encounter high packet loss. In many broadcast applications, lost packets are not retransmitted. Hence the authentication protocol should tolerate high levels of packet loss.

- *Scalability.* Broadcast applications have a potentially large number of receivers. The authentication information should be independent of the number of receivers.

- *Small size of authentication information.* Since the receiver authenticates individual messages instantly, each message carries authentication information, hence a viable scheme should have a low communication overhead.

Researchers proposed a number of schemes for broadcast authentication [4, 6, 16, 17, 23, 25]. Unfortunately most previously proposed systems [6, 16, 17, 23, 25] cannot simultaneously support both real-time authentication and perfect robustness to packet loss. One approach that supports both real-time authentication and robustness to packet loss [4] does not scale well to large number of receivers, as the size of authentication information increases as the number of receivers increases. Using a new construction, we design the BiBa broadcast authentication protocol from the BiBa signature scheme, that satisfies all the above desired properties, except that the sender overhead to generate the authentication information is in general higher than for previous approaches (although it can be parallelized). In addition, the BiBa broadcast authentication protocol requires that the sender and receiver are weakly time synchronized.[1]

Similar to the MicroMint payment scheme by Rivest and Shamir [21], the security of the BiBa signature comes from the difficulty of finding k-way collisions for a one-way function. The main difference, however, is the security assumption: MicroMint assumes that the bank has more computational resources than an adversary, but BiBa enjoys exponentially increasing security such that it is secure even if the signer only has modest computation resources.[2]

Our Contributions

We propose the BiBa signature scheme, a new one-time signature scheme based on one-way functions without trapdoors. The BiBa signature exploits the birthday paradox to achieve efficiency and security.

BiBa provides a more compact signature and is faster to verify than previous schemes. The public verification key can be large, but if off-line dissemination of the public key is possible, the BiBa signature offers super-fast verification. We believe that it provides one of the fastest signature verifications today.

We design a broadcast authentication protocol based on the BiBa signature scheme. It was an open problem to design a broadcast authentication system that can simultaneously support efficient real-time transmission and efficient authentication, offer perfect robustness to packet loss, and scale perfectly with respect to the number of receivers. Our construction is general and also applies to other signature schemes based on one-way functions without trapdoors, e.g. the k-times signature [23].

[1]In contrast to TESLA, the authentication in BiBa is instant and does not depend on the time synchronization error. However, a large synchronization error results in a higher memory requirement of the receiver.

[2]MicroMint gains an additional computational advantage because the bank can pre-compute coins, and an adversary has a small, limited time to forge coins. Hence MicroMint requires very loose time synchronization.

2. THE BIBA SIGNATURE SCHEME

In this section we first introduce the notion of SEALs, then we give a simple example to motivate the key intuition of BiBa, and finally we present the full-fledged scheme BiBa signature scheme.

We use the following notation in the rest of the paper. F and F' represent two pseudo-random functions (PRF)[7], $F : \{0,1\}^{m_2} \times \{0,1\}^{m_1} \to \{0,1\}^{m_2}$
$F' : \{0,1\}^{m_1} \times \{0,1\}^{m_1} \to \{0,1\}^{m_1}$.
H is a hash function in the random oracle model[1]. G represents a hash function family in the random oracle model and $G_h : \{0,1\}^{m_2} \to [0, n-1]$ is an instance in the hash function family G selected with an indicator h.

2.1 The SEALs

The signer precomputes values that it subsequently uses to generate BiBa signatures. These values are random numbers generated in a way that the receivers can instantly authenticate them with the *public key* (which is sometimes referred to as *public validation information* in this context). We call these precomputed values *SEALs*, short for *SElf-Authenticating vaLues*.[3] The property that we need for SEALs is that the verifier can efficiently authenticate the SEAL based on the public key, and that it is computationally infeasible for an adversary to find a valid SEAL given a public key. The simplest approach is to use the PRF F as a commitment scheme. Given a SEAL s, the public key is $f_s = F_s(0)$. If the verifier learns f_s in an authentic fashion, it can easily authenticate s by verifying $F_s(0) = f_s$. In BiBa the signer needs multiple SEALs, so a public key could consist of multiple commitments.

Another alternative for SEAL authentication is a Merkle hash tree (so the SEALs would be the leaf nodes of the tree and the public key is the root node of the tree) [12]. We discuss this approach in Appendix B.

In the case of broadcast authentication, we describe efficient methods to generate and authenticate SEALs in Section 3. For now we simply assume that the signer has t precomputed SEALs $s_1, s_2, \ldots, s_t$, and receivers know the authentic commitment $F_{s_i}(0)$ of each SEAL so they can efficiently authenticate each SEAL they receive.

2.2 The Key Intuition

BiBa stands for *Bins and Balls* signature — a collision of balls under a hash function in bins forms the signature. BiBa exploits the birthday paradox such that the signer has many balls to throw into the bins which results in a high probability to find a signature, but an adversary has few balls so it has a low probability to forge a signature. We illustrate the BiBa signature with a simplified example.

Signature Generation

To sign a message m, the signer first computes the hash $h = H(m)$. The signer then computes the hash function G_h (as we describe at the beginning of this section) to all the SEALs $s_1, \ldots, s_t$. The signer looks for a two-way collision of two SEALs: $G_h(s_i) = G_h(s_j)$, with $s_i \neq s_j$. The pair $\langle s_i, s_j \rangle$ forms the signature. Figure 1 shows an example.

[3]The name SEAL is already used for the stream cipher developed by Rogaway and Coppersmith. We overload the term in this work because a seal nicely describes the properties that a signer can easily generate it, that it is hard to forge, and that a verifier can easily authenticate it.

It is now clear why the BiBa acronym stands for Bins and Balls signature: the bins correspond to the range of the hash function G_h, and the balls correspond to the SEALs.

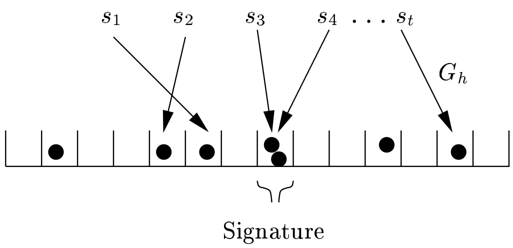


Figure 1: Basic BiBa scheme

We exploit the asymmetric property that the signer has more SEALs than the adversary, and hence the signer can easily generate the BiBa signature with high probability. On the other hand, an adversary only knows the few disclosed SEALs and hence has a low probability to find a valid BiBa signature.

Signature Verification

The verifier receives message M and the BiBa signature $\langle s_i, s_j \rangle$. We assume for now that the verifier has an efficient method to authenticate the SEALs s_i, s_j. To verify the BiBa signature the verifier computes $h = H(m)$, checks that $s_i \neq s_j$, and $G_h(s_i) = G_h(s_j)$.

Note that the verification is very light-weight: Without considering the SEAL authentication, the verification only requires one hash function computation and two hash function computations.

Security of This Approach

Assume the signer has t SEALs and the range of the hash function G_h is $[0, n-1]$. Given a message m, the probability of finding a BiBa signature is equal to the probability of finding at least one two-way collision, i.e. at least two balls end up in the same bin, when throwing t balls uniformly randomly into n bins. The probability of at least one collision P_C is easy to compute:

$$P_C = 1 - \prod_{i=1}^{t-1} \frac{n-i}{n} \approx 1 - e^{\frac{t(t+1)}{2n}}$$

Figure 2 shows the probability of at least one collision when throwing t balls into 762460 bins.

The security of the BiBa signature comes from the fact that the adversary has few SEALs and hence has a small probability to find a collision. For example, if the signer has 1200 SEALs, as marked with the letter A in Figure 2, it has a 61% probability of finding a signature after one try (one two-way collision after throwing 1200 balls into 762460 bins) for a given a message. If we assume that an adversary only knows 10 SEALs (which it learned from 5 BiBa signatures), it has a $2^{-13.9}$ probability to find a collision (forge a signature) after one try. Figure 2 shows the adversary's probability, marked with the letter B.

2.3 BiBa Extensions

The first way to increase the security is to increase the number of SEALs and bins, but that approach would increase the size of the public key.

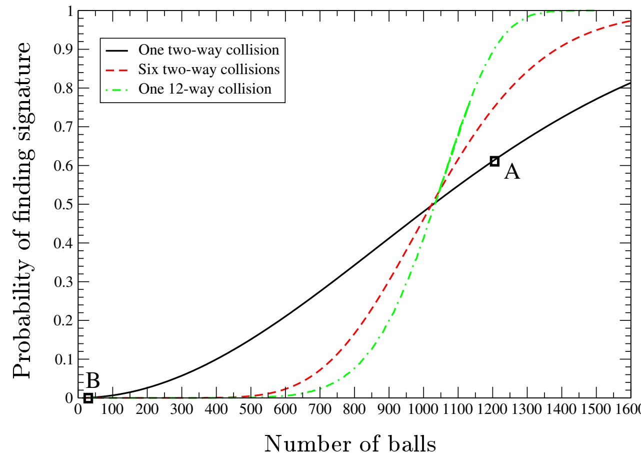


Figure 2: Probability of finding a signature for three cases. The solid line shows the probability for finding a two-way collision when throwing x balls into 762460 bins. The dashed line shows the probability of finding six two-way collisions when throwing x balls into 236650 bins. The dot-dashed line shows the probability of finding a six-way collision when throwing x balls into 222 bins. We selected the number of bins such that the signer has a 50% probability of finding a signature with 1024 SEALs after one try.

A better approach to increase the security is to use multiple two-way collisions to generate a signature. For example, a signature on message m, with $h = H(m)$, would consist of z two-way collisions. The signature is composed of z pairs of SEALs $\langle S_{a_1}, S_{b_1} \rangle, \ldots, \langle S_{a_z}, S_{b_z} \rangle$, with all SEALs distinct and $G_h(S_{a_i}) = G_h(S_{b_i})$ $(1 \leq i \leq z)$. Figure 2 shows the probability of finding six two-way collisions with $n = 236650$ bins, varying the number of SEALs.

The third way to increase security is to require multi-way collisions, instead of two-way collisions. If the BiBa signature requires a k-way collision, the BiBa signature of message m (with $h = H(m)$) is $\langle S_{x_1}, \ldots, S_{x_k} \rangle$, where all SEALs $S_{x_1}, \ldots, S_{x_k}$ are distinct and collide under G: $G_h(S_{x_1}) = \ldots = G_h(S_{x_k})$. Figure 2 shows the probability of finding a six-way collision with $n = 222$ bins, varying the number of SEALs. The figure shows clearly that this approach is better than using six two-way collisions, because the probability drops off faster for an adversary that has fewer SEALS.

The fourth way we attempted to improve the security is to use a multi-round scheme, where only the bins that have a k_1-way collision in the first round proceed as balls into the next round. Intuitively, multi-round schemes may seem to improve the security. However, we show in Appendix A that one-round schemes are as secure as multi-round schemes.

2.4 The BiBa Signature Scheme

We now describe the BiBa signature scheme in more detail. To sign message m the signer computes the m_3 bit long hash $h = H(m|c)$, where c is a counter value that the signer increments if it cannot find a signature. The signer has t SEALs (each is m_2 bits long), and maps them with the hash function G_h into n bins. Any k-way collision of SEALs forms the signature.

Verification is straightforward. We assume that the verifier knows the BiBa parameters k and n, the hash function H, and the hash function family G.

Assume the verifier receives message m and BiBa signature $\langle S_{x_1}, \ldots, S_{x_k}, c\rangle$. First, the verifier verifies that all k SEALs are distinct and authentic. Next, the verifier computes $h = H(m|c)$, and accepts the signature if all $G_h(S_{x_i})$ (for $1 \le i \le k$) are equal.

2.5 Security Considerations

Given a number of disclosed SEALs, we can derive the probability that an adversary can find a valid BiBa signature using standard combinatorial techniques. In Appendix A we derive that a tight upper bound of the probability P_f of the adversary to forge a signature after a single trial is

$$P_f = \frac{\binom{r}{k}(n-1)^{r-k}}{n^{r-1}}$$

where r is the number of SEALs that the adversary knows.

Clearly the more SEALs the adversary has, the higher the probability that it can forge a BiBa signature. The security of the BiBa signature scheme relies on the fact that the signer knows many more SEALs than an adversary. We leverage the birthday paradox threshold, also known as the *birthday bound*. Intuitively, when we throw t balls uniformly randomly into n bins, when the number t is below the birthday bound (approximately $\sqrt{2n}$), two-way collisions are rare. But as t grows larger than the birthday bound, the number of collisions increases rapidly.

k	n	P_f	k	n	P_f
2	762460	$2^{-19.5403}$	13	192	$2^{-91.0196}$
3	15616	$2^{-27.8615}$	14	168	$2^{-96.1001}$
4	3742	$2^{-35.6088}$	15	151	$2^{-101.3377}$
5	1690	$2^{-42.8912}$	16	136	$2^{-106.3119}$
6	994	$2^{-49.7855}$	17	123	$2^{-111.0802}$
7	672	$2^{-56.3539}$	18	112	$2^{-115.7250}$
8	494	$2^{-62.6386}$	19	104	$2^{-120.6079}$
9	384	$2^{-68.6797}$	20	96	$2^{-125.1143}$
10	310	$2^{-74.4851}$	21	89	$2^{-129.5147}$
11	260	$2^{-80.2237}$	22	83	$2^{-133.8758}$
12	222	$2^{-85.7386}$	23	78	$2^{-138.2788}$

Table 1: The security of some BiBa instances. The signer knows $t = 1024$ SEALs and the adversary has $r = k$ SEALs.

We assume the signer has $t = 1024$ SEALs, and the attacker has $r = k$ SEALs which is the number of SEALs disclosed after the signer signs one message. Let P_S denote the probability that the signer can find a signature after one trial with t SEALs, and P_f denotes the probability that an adversary can forge a signature after one trial knowing r SEALs. A high value for P_S means the signer is likely to produce a signature on the message and hence signing is efficient. For the remainder of this paper we set $P_S = 0.5$, such that the signer can find a signature after 2 tries on average. A low value for P_f means that it is unlikely for an attacker to produce a valid signature, and indicates the security of the scheme. Table 1 shows the P_f value for different instances with varying parameters n and k. In Section 5 we discuss methods on how to choose the BiBa parameters, and we discuss the resulting communication and computation costs. We would like to point out the high level of security BiBa achieves with only a few collisions. Based on the requirements of the NESSIE project [15], a BiBa signature scheme that uses an 11-way collision would provide sufficient security.

An adversary has two main ways to collect SEALs to forge signatures. First, the adversary simply collects SEALs disclosed in signatures generated by the signer. Second, the adversary tries to find SEALs by brute-force computation to invert the PRF F used to authenticate the SEALs. In our analysis, however, we assume that the latter attack is impractical, such that the adversary only knows the SEALs that the signer discloses with a signature.

3. BIBA BROADCAST AUTHENTICATION PROTOCOL

In this section we describe how we use the BiBa signature to design the BiBa broadcast authentication protocol.

A broadcast authentication protocol requires that each receiver can verify that the data originates from the sender. An obvious approach is for the sender to compute a BiBa signature on each message it broadcasts. Since the sender can only disclose a small number of SEALs, it could only sign a small number of messages (given a public key which commits to a fixed number of SEALs). For a viable broadcast authentication protocol, however, the sender needs to authenticate a potentially infinite stream of messages. So we construct a protocol that replenishes the SEALs disclosed with each signature. In a straightforward approach, the sender adds a new commitment (for each SEAL that it discloses) to the packet, and includes all the new commitments in the signature. This approach doubles the size of the signature and is not robust to packet loss. We now present a better approach for constructing the SEALs. We will not review signature generation and verification in this section, since it is the same as in the BiBa signature scheme that we describe in Section 2.4.

One-way SEAL Chains

For our authentication protocol, we need a method such that the receiver can instantly authenticate the SEALs when it receives them, and that the SEALs are automatically replenished. We use *one-way chains* to achieve the self-authenticating property of SEALs and for replenishment. One-way chains are used in many schemes, for example by Lamport in a one-time password system [11], and the S/Key one-time password system [8].

We use the the PRF F to generate the one-way SEAL chains, and the PRF F' to generate a one-way salt chain. The sender first generates the *one-way salt chain* of length l, $\{K_i\}_{1 \le i \le l}$, using the PRF F' as follows: the sender randomly selects K_l (of length m_1 bits): $K_l \xleftarrow{R} \{0,1\}^{m_1}$, and then recursively computes all other salts: $K_i = F'_{K_{i+1}}(0)$ $(1 \le i < l)$.

The sender then generates a set of *one-way SEAL chains*, $\{S_{\langle i,j\rangle}\}_{1 \le i \le t, 1 \le j \le l}$, where $S_{\langle i,\_\rangle}$ forms a one-way chain as Figure 3 shows. The SEAL chains are constructed as follows. The sender first randomly selects all the seed SEAL values $S_{\langle \_,l\rangle}$ of length m_2 bits: $S_{\langle i,l\rangle} \xleftarrow{R} \{0,1\}^{m_2}$ $(1 \le i \le t)$. The sender then computes all other SEAL values recursively: $S_{\langle i,j\rangle} = F_{S_{\langle i,j+1\rangle}}(K_{j+1})$ $(1 \le j < l)$. Note the way we use the salts of the one-way salt chain to derive the SEAL values, such that an attacker first would need to find a pre-image of the salt of the one-way salt chain before it can try to find pre-images for the SEAL chains. We chose this specific construction to allow for relatively compact SEALs, while the longer salts mitigate attacks to find other pre-images for the SEALs by pre-computation. However, if the SEALs are long enough to prevent such attacks, the one-way salt chain may not be necessary.

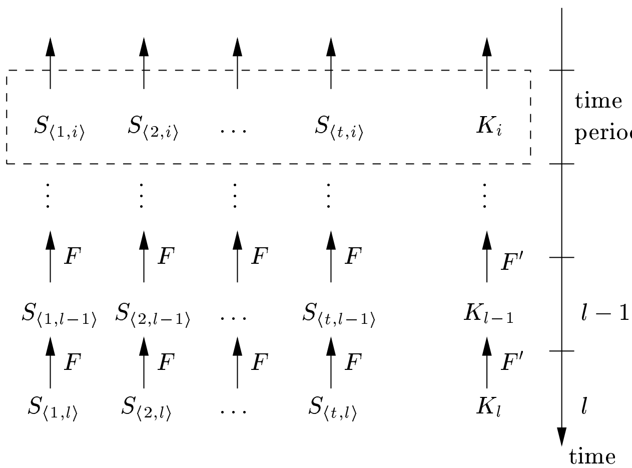


Figure 3: Using one-way chains to construct SEAL

The sender divides the time up into time periods of equal duration T_d. In each time period i, the SEALs $S_{\langle \_,i\rangle}$ and the salt K_i are *active*. Figure 3 shows the time periods and the corresponding *active SEALs* and *active salt*. As time advances an entire row of SEALs expires and a new row becomes active. The sender publishes each salt at the beginning of the time period when it becomes active, but the sender only discloses the active SEALs of a time period that are part of a BiBa signature.

To bootstrap a new receiver we assume for now that the sender sends it all the SEALs and the salt of a previous time period over an authenticated channel. We present extensions that provide more efficient receiver bootstrapping in Section 5. It is clear that a receiver who knows all the authenticated SEALs and salt of a time period can authenticate SEALs and salts of later time periods. For example, assume the receiver knows the authentic salt K_i of time period i. When the receiver receives K_{i+1} of the following time period the receiver authenticates it by verifying $K_i \stackrel{?}{=} F'_{K_{i+1}}(0)$. The receiver authenticates SEALs by following the one-way SEAL chain back to a SEAL that it knows is authentic.

Security Condition

To prevent an adversary from forging for a signature, we need to ensure that an adversary knows few active SEALs. Hence, when the receiver receives a packet, the receiver has to be certain that an adversary could only know a small number of SEALs. The receiver can verify such a condition if it is time synchronized with the sender and knows the sending schedule of packets. We refer to TESLA for more details on time synchronization [16]. Assuming a maximum time synchronization error of δ between the sender and the receivers, the sender is limited to sign $\lfloor r/k \rfloor$ messages within time δ, where r is the maximum number of active SEALs that the adversary can know, and k is the number of SEALs revealed in a signature. When the receiver gets a packet it needs to verify that the sender did not yet disclose more than r active SEALs. Because of the one-way SEAL chains, SEALs of one time period also disclose SEALs from previous time periods. Hence we require that the sender does not use a BiBa instance for time δ after it disclosed k SEALs of that instance. To send continuously, the sender needs to use multiple BiBa broadcast authentication instances in a round-robin fashion.

4. BIBA BROADCAST PROTOCOL EXTENSIONS

We briefly present two extensions to the BiBa broadcast protocol. An optimal protocol would satisfy the following three properties: low receiver computation overhead (as low as the BiBa signature protocol), low communication overhead (only the disclosed SEALs are in the packet), and perfect robustness to packet loss. Unfortunately, we could not find such a protocol and we leave its quest to future research. However, we can achieve two out of the three properties. The standard BiBa broadcast authentication protocol we describe in the previous section has low communication overhead, perfect robustness to packet loss, but requires more receiver computation overhead than the standard BiBa signature protocol (to verify the authenticity of the SEALs). In this section, we propose the extension A and B. Extension A does not tolerate packet loss, and extension B has a higher communication overhead. Hybrid schemes exist, but we do not describe them here. The difference among the three protocols is how they manage the SEALs.

Extension A

Extension A provides low receiver computation overhead and low communication overhead, but it does not tolerate packet loss. The basic BiBa broadcast authentication protocol has a high receiver computation overhead because the majority of the SEALs are never used in a signature, so to authenticate a SEAL the receiver needs to recompute many SEALs in a one-way SEAL chain until it reaches a previously stored SEAL. We solve this problem in extensions A and B by using every SEAL of each one-way SEAL chain in a BiBa signature.

In this scheme we use the concept of *SEAL boundary*, which Figure 4 depicts. The SEALs above the boundary are disclosed, and they serve as commitment to the SEALs on the other side of the boundary. The sender and receiver always know the current SEAL boundary. The sender only uses SEALs that are directly adjacent to (below) the boundary. After each BiBa signature the sender and receiver extend the SEAL boundary past the newly disclosed SEALs.

This scheme would not be secure if an adversary could slow down the data traffic to the receiver, and collect enough packets such that it knows a large number of SEALs on the lower side of the perceived SEAL boundary of the receiver. This large number of SEALs would enable the adversary to spoof subsequent data traffic, because the adversary continuously receives fresh SEALs that the sender discloses. This illustrates that the sender and receiver need to be time synchronized, such that the receiver knows the sending schedule of the packets.

As an additional security measure, the sender can also sign

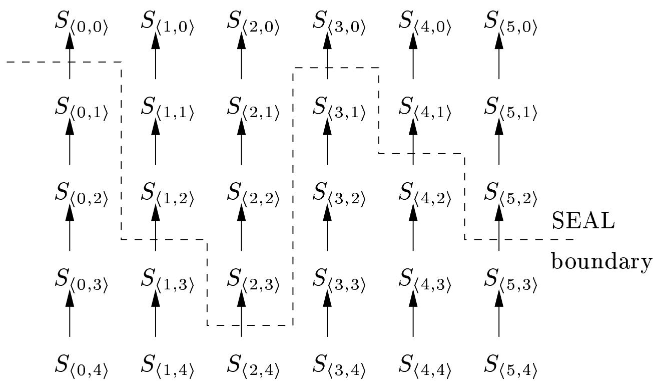


Figure 4: Using one-way chains to construct SEAL

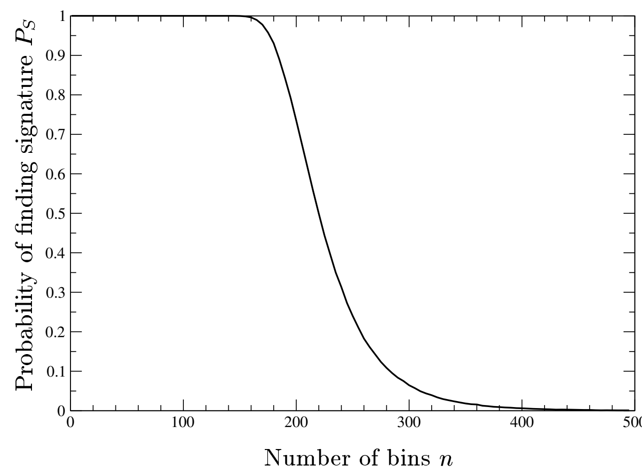


Figure 5: Probability of finding a twelve-way collision when throwing 1024 balls into x bins.

all the SEALs directly above the current SEAL boundary with the message signature. This mechanism would ensure the receiver that it knows the correct SEAL boundary.

Extension B

Extension B is similar to extension A, but it tolerates packet loss. Extension A does not tolerate packet loss because the receiver needs to know which SEALs the sender discloses so it can update the SEAL boundary. To improve the tolerance to packet loss, we add SEAL boundary information to packets, which increases the communication overhead. This approach allows us to trade off robustness to packet loss with communication overhead.

Two main methods exist to encode the SEAL boundary in packets: *absolute encoding* or *relative encoding*. The absolute encoding sends the index of each SEAL of the SEAL boundary in the packet. For instance the SEAL boundary in Figure 4 is $\langle 0, 2, 3, 0, 1, 2\rangle$. A relative encoding only communicates the change of the SEAL boundary with respect to a previous boundary.

As in extension A, the receiver needs to know the exact sending rate of messages, to verify that the SEAL boundary always grows by the number of SEALs that the sender discloses. To prevent that an attacker collects more than r SEALs, the receiver needs to receive at least one packet every ν packets (where $\nu = \lfloor r/k \rfloor$). Hence this scheme does not tolerate more than $\nu - 1$ consecutively lost packets. Otherwise, an attacker could collect SEALs during a long period of packet loss, and forge subsequent packets by claiming a bogus SEAL boundary.

5. PRACTICAL CONSIDERATIONS

In this section we first discuss how to derive the BiBa parameters from a given set of system requirements, and we analyze the impact of BiBa parameters on the overhead and security. Next, we look at the overhead of signature generation and verification. We will then discuss a concrete example on how to construct a viable broadcast authentication scheme and discuss the performance.

Selection of BiBa Parameters

We assume that the sender has $t = 1024$ SEALs. Let P_f denote the probability that an attacker can find a signature with one trial of one message knowing at most r SEALs. The security parameter is generally expressed as the expected number of hash function operations that an adversary has to perform to forge a signature [15]. For BiBa, the minimum number of hash function operations to forge a signature is $2/P_f$, for simplicity we use $1/P_f$.

Let P_S denote the probability that the sender can find a signature in one trial. The expected number of tries that the sender performs to find a signature is $1/P_S$. Without loss of generality we set $P_S = 0.5$.

To achieve good security, the sender can disclose approximately up to $\gamma = 10\%$ of the SEALs. Each signature reveals k SEALs. The sender knows t SEALs, so it can produce $\nu = \lfloor \gamma \cdot t/k \rfloor$ signatures in a time period. As we discuss in Section 3, the sender needs to wait for time δ until it can disclose the SEALs of the next time period. Hence it needs multiple BiBa instances, if it wants to send more than ν messages per time period δ. Given the packet sending rate β, the number of BiBa instances needed is $\lceil \delta\beta/\nu \rceil$.

We now discuss how we choose n and k. The choice of k directly determines the signature size. We can derive the number of bins n from k and the probability P_S that the sender finds a signature after one trial. Figure 5 shows how P_S decreases as we increase n.

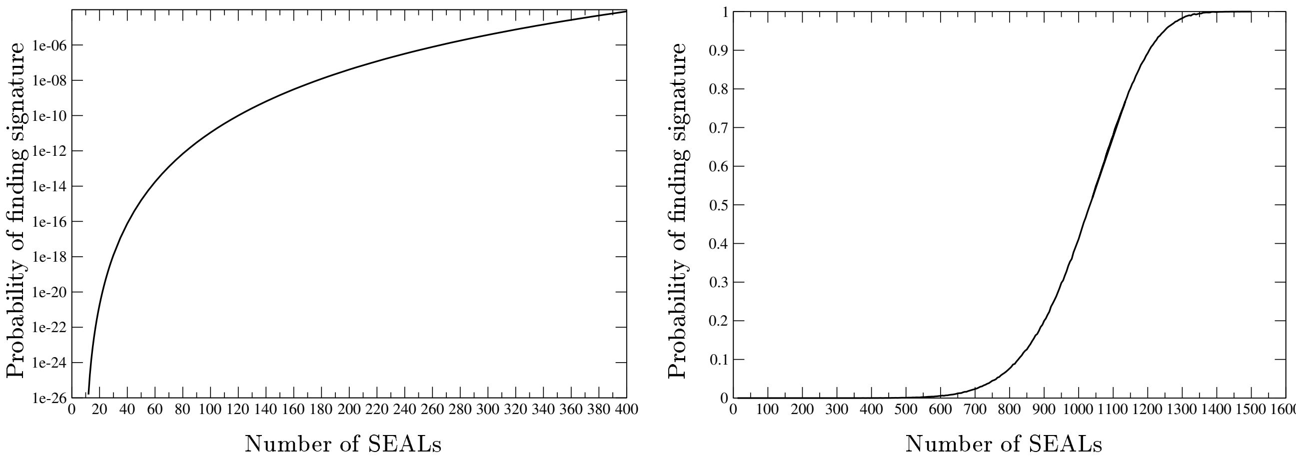
Once we fix n and k we can derive the number of SEALs that the sender can disclose such that the adversary has at most a probability of P_f to forge a signature. Figure 6 depicts the probability distribution to find a signature given a certain number of SEALs. As we can see in Figure 6(a), P_f quickly decreases as the sender decreases the number of SEALs it discloses. If P_f is too high (insufficient security) for a k-way collision, we need to increase k.

BiBa Overhead

Table 2 lists the computation, memory, and communication overhead of the sender and the verifier during the precomputation, signature generation, and signature verification phases, where T_F, T_G, and T_H denote the time to compute the functions F, G, H, respectively.

Example: Real-time stock quotes

Consider a real-time stock quote broadcasting system. The main requirement is a low authentication delay for the real-time data, hence buffering on either the sender or receiver side is not an option. Another requirement is the robustness



(a) $12 \leq x \leq 350$

(b) $12 \leq x \leq 1600$

Figure 6: Probability of finding a signature given x SEALs. These probabilities are for the scheme $k = 12, n = 222$.

	Computation	Memory
Precomputation	$l(t+1)T_F$	$l(m_1 + t \cdot m_2)$
Signature Generation	$(t \cdot T_G + T_H)/P_S$	$l(m_1 + t \cdot m_2)$
Signature Verification	$2 \cdot k \cdot T_G + T_H$	$m_1 + (k+n) \cdot m_2$

Table 2: BiBa Overhead. The salts are m_1 bits long, and the SEALs are m_2 bits long. The communication overhead (signature size) is $k \cdot m_2$ ($+m_1$ if we also send the salt).

to packet loss, so receivers can instantly and efficiently authenticate each message they receive, despite previously lost packets. To the best of our knowledge, no previous protocol satisfactorily addresses these requirements.

We assume the following system requirements. Receivers are time synchronized with the sender, with a maximum time synchronization error of ten seconds ($\delta = 10$s). The sending rate is approximately 10 packets per second ($\beta = 10$ p/s). The sender has $t = 1024$ SEAL chains. We set the security parameter $1/P_f = 2^{58}$ (the attacker needs to perform 2^{58} hash function computations within time δ to forge a signature). We set $\gamma = 1/16$, so the adversary can know up to $r = t\gamma = 64$ active SEALs.

For this example, we set $m_1 = 128, m_2 = 64, k = 16$. We then compute n to get P_S close to 50% and compute the corresponding P_f assuming that the adversary knows 64 SEALs. From Table 1 we pick $n = 168$, and the resulting forging probability is $P_f = 2^{-58.03}$. The signature size is about 128 bytes.

Each signature discloses 16 SEALs, hence after 4 packets an attacker knows at most 64 SEALs. Each row of the SEAL chains is active for 400ms (10 packets/s and 4 packets sent per row). Because $\delta = 10s$ is the maximum time synchronization error, we cannot disclose any SEALs of the next row for 10 seconds. Otherwise we would disclose more SEALs of the previous rows which the adversary could use to forge a signature. This requires that we use 25 instances of the BiBa scheme in parallel in a round-robin fashion.

The sender overhead to generate a signature is approximately $(1024 \cdot T_G + T_H)/P_S$. On a 800 MHz Pentium III the sender can compute $\approx 10^6$ MD5 hash function evaluations per second [19] (uniprocessor, software-based implementation of MD5). On this architecture, generating one BiBa signature takes approximately 2 ms, about 5 times faster than to generate a 1024-bit RSA signature using the OpenSSL library. BiBa enjoys a linear speedup for multiple processors, which a hardware implementation can easily exploit. With maximum parallelization, generating a BiBa signature only requires two sequential hash function computations.

Signature verification (excluding the verification of the authenticity of the SEALs) only requires 17 hash function evaluations. However, the SEAL verification is about 256 PRF evaluations on average. This is because the 1024 active SEALs of one time period are amortized only by 4 packets. On a 800 MHz Pentium III the sender can compute $\approx 5 \cdot 10^6$ RC5 function evaluations per second [20] (uniprocessor, software-based implementation of RC5). On this architecture, verification takes on the order of 50μs, which is about 20 times faster than verifying a 1024-bit RSA signature.

Decreasing the security requirement and increasing the number of disclosed SEALs would greatly diminish this number (e.g. If we allow the adversary to know 128 SEALs which would result in $P_f = 2^{-41.2}$, then the verifier only needs to perform 128 PRF evaluations on average to authenticate the SEALs). Using either extension A or B would reduce the verification overhead to 17 hash function and 16 PRF computations, however with the tradeoffs we describe in Section 4.

Efficient Public-Key Distribution

Sending the public key to all receivers is a potential bottleneck. In the schemes we discuss in this paper, the public key size is on the order of 10 Kbyte for each BiBa instance. We now present a trick that makes public key distribution efficient for the sender, but requires a longer time to bootstrap receivers. The intuition is that receivers can collect SEALs while they receive signed messages, and reconstruct the one-way SEAL chains and the one-way salt chain. Periodically, the sender broadcasts a message containing the hash of all SEALs and the salt of one time period, signed with a traditional digital signature scheme, for example RSA [22]. Once the receiver collects all SEAL chains, it can authenticate them with the digital signature and authenticate subsequent traffic. This assumes that the receiver is already time synchronized with a maximum time synchronization error δ. The well-known coupon collector problem predicts how long the receiver needs to wait: After collecting $t \cdot \log(t)$ random SEALs, it has one SEAL of each one-way chain with high probability, where t is the number of SEAL chains. In the schemes we consider in this paper $t = 1024$, hence the receiver needs to collect about 7098 SEALs. In our first example, the sender discloses 64 SEALs in each time period, so the receiver needs to collect SEALs during 110 time periods.

6. CONCLUSION

The BiBa signature uses the birthday paradox to construct a digital signature scheme from a one-way function without a trapdoor. A BiBa signature is a k-way collision of input values under a hash function in the random oracle model. In comparison to previous approaches the BiBa signature offers smaller signature size, and a smaller verification overhead. On the downside, the public key is larger and the signature generation overhead is higher than for previous approaches.

The BiBa one-time signature is particularly useful in settings where the signer can send the public key to the verifier efficiently, or where the verifier is constrained on computation power. Furthermore, the BiBa signature generation enjoys a linear speedup on multiprocessor systems until signature generation and verification only require two sequential hash function evaluations.

Broadcast authentication remains an important problem. As far as we know, no previous protocol could provide perfect robustness to packet loss, instant authentication (without sender- or receiver-side buffering), efficient verification, and scalability to a large number of receivers. We propose a new construction to achieve perfect loss robustness for signature schemes based on one-way functions without trapdoors. By combining our new construction with the BiBa signature scheme, we obtain the BiBa broadcast authentication protocol.

7. ACKNOWLEDGMENTS

This research started at Digital Fountain: I am indebted to Mike Luby for his encouragement to improve broadcast authentication schemes, and to Hugo Krawczyk for his work on new schemes which inspired the development of BiBa and for his helpful feedback. I would like to thank Avi Wigderson for his help throughout this project. In particular, I would like to thank Avi for pointing out that one-round schemes are at least as good as multi-round schemes. I would also like to thank Dawn Song for her help, and especially for her contribution to Appendix B. Special thanks also go to Ran Canetti for his helpful comments. I would also like to thank Nikita Borisov, Monica Chew, Yongdae Kim, Michael Mitzenmacher, Amin Shokrollahi, Gene Tsudik, Doug Tygar, and David Wagner for helpful feedback and discussions.

8. REFERENCES

[1] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Victoria Ashby, editor, *1st ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, November 1993. ACM Press. Appeared also (in identical form) as IBM RC 19619 (87000) 6/22/94.

[2] D. Bleichenbacher and U. M. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In Y. G. Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 75–82.

[3] G. Brassard, editor. *Advances in Cryptology – CRYPTO '89*, number 435 in Lecture Notes in Computer Science, Santa Barbara, CA, USA, 1990.

[4] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, Mar. 1999.

[5] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In Brassard [3], pages 263–277.

[6] R. Gennaro and P. Rohatgi. How to sign digital streams. In B. S. Kaliski, Jr., editor, *Advances in Cryptology – CRYPTO '97*, number 1294 in Lecture Notes in Computer Science, pages 180–197.

[7] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[8] N. Haller. The S/Key one-time password system. In D. N. G. Chair) and R. S. P. Chair), editors, *Symposium on Network and Distributed Systems Security*, San Diego, California, Feb. 1994.

[9] L. Lamport. Discussion with Whitfield Diffie. `http://research.compaq.com/SRC/personal/lamport/pubs/pubs.html#dig-sig`, 1975.

[10] L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.

[11] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[12] R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, Apr. 1980.

[13] R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO '87*, number 293 in Lecture Notes in Computer Science, pages 369–378, Santa Barbara, CA, USA, 1988.

[14] R. C. Merkle. A certified digital signature. In Brassard [3], pages 218–238.

[15] Nessie: New European schemes for signatures, integrity, and encryption. `http://www.cryptonessie.org`, 1999.

[16] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and secure source authentication for multicast. In *Symposium on Network and Distributed Systems Security (NDSS 2001)*, San Diego, CA, Feb. 2001. Internet Society.

[17] A. Perrig, R. Canetti, D. Tygar, and D. Song. Efficient authentication and signature of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 2000.

[18] M. O. Rabin. Digitalized signatures. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[19] Ron Rivest. The MD5 message-digest algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force, April 1992.

[20] Ron Rivest. The RC5 encryption algorithm. In Anderson, editor, *Proceedings of the 1st International Workshop on Fast Software Encryption*, volume 809 of *Lecture Notes in Computer Science*, pages 86–96, 1994. Springer-Verlag, Berlin Germany.

[21] R. L. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. In M. Lomas, editor, *Security Protocols—International Workshop*, volume 1189 of *Lecture Notes in Computer Science*, pages 69 – 88, Cambridge, United Kingdom, Apr. 1997. Springer-Verlag, Berlin Germany.

[22] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[23] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet. In *6th ACM Conference on Computer and Communications Security*, pages 93–100, Singapore, Nov. 1999. ACM Press.

[24] R. von Mises. Über Aufteilungs- und Besetzungswahrscheinlichkeiten. *Revue de la Faculté des Sciences de l'Université d'Istanbul*, 4:145—163, 1939.

[25] C. K. Wong and S. S. Lam. Digital signatures for flows and multicasts. In *IEEE ICNP '98*, 1998.

APPENDIX

A. ONE-ROUND BIBA IS AS SECURE AS MULTI-ROUND BIBA

We illustrate a multi-round scheme with a concrete example. Figure 7 illustrates this approach for two rounds. In the first round, we look for two-way collisions under a hash function G_h. The indices of the bins that have at least a two-way collision in the first round are used as balls in the second round, and a three-way collision is required under $\hat{G}_h$. The indices that participate in the three-way collision in the second round and the SEALs necessary to verify that each index corresponds to a bin with the two-way collision in the first round, form the BiBa signature: $\langle S_{x_1}, \ldots, S_{x_6} \rangle$.

To verify the BiBa signature $\langle S_{x_1}, S_{x_2}, S_{x_3}, S_{x_4}, S_{x_5}, S_{x_6} \rangle$ of message m (with $h = H(m)$), the verifier verifies the following conditions:

1. All six SEALS S_i are distinct and authentic
2. Throwing all 6 SEALs into n_1 bins results in 3 two-way collisions in bins with indices b_1, b_2, b_3. These indices form a three-way collision under $\hat{G}$: $\hat{G}_h(b_1) = \hat{G}_h(b_2) = \hat{G}_h(b_3)$.

We now sketch a proof that a one-round BiBa signature are as good as a multi-round signature scheme. Please consult the full version of this paper for an extended proof.

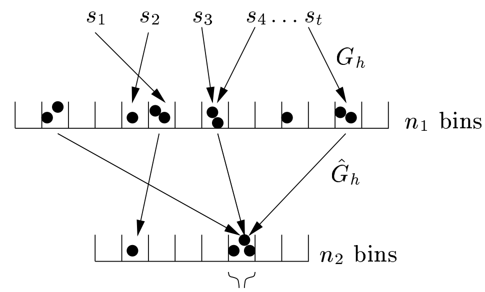


Figure 7: Signature scheme with two rounds

LEMMA 1. *A one-round BiBa signature is asymptotically as secure as a two-round BiBa signature.*

PROOF SKETCH. We start out by showing that a one-round scheme offers the same security as the corresponding two-round scheme with the same signature size. We then use an inductive argument to show that an $n+1$-round scheme is no more secure than an n-round scheme.

Some of the following formulas are due to von Mises' seminal article on the occupancy probabilities in bins and balls models [24]. The expected number of bins that contain exactly k balls (k-way collisions) after throwing t balls into n bins is:

$$E[k] = \frac{\binom{t}{k}(n-1)^{t-k}}{n^{t-1}} \tag{1}$$

We use the following inequalities:

$$E[k] = \frac{\binom{t}{k}(n-1)^{t-k}}{n^{t-1}} < \frac{\binom{t}{k}(n)^{t-k}}{n^{t-1}} = \frac{\binom{t}{k}}{n^{k-1}} \tag{2}$$

$$\frac{\binom{t}{k}}{n^{k-1}} = \frac{t!}{(t-k)!k!n^{k-1}} < \frac{t^k}{k!n^{k-1}} \tag{3}$$

We write $P_k(x)$ for the probability that exactly x k-way collisions occur. In the settings that we study, we have $P_k(1) > P_k(2)$, and in particular for the adversary who tries to forge a signature we have $P_k(1) \gg P_k(2) \gg P_k(3)$. Similarly, we have $P_k(1) \gg P_{k+1}(1)$ since the signer can barely find a k-way collision, and for the adversary we have $P_{k+1}(1) = 0$ because it only knows k SEALs that were disclosed in a signature. From the first equality of equation 1 we approximate the probability to find a valid signature

$$\begin{aligned} P_f &\approx P_k(1) + P_k(2) + \cdots \\ &= E[k] - P_k(2) - 2P_k(3) - \cdots \\ &\approx E[k] \end{aligned} \tag{4}$$

$$P_f \approx \frac{\binom{t}{k}(n-1)^{t-k}}{n^{t-1}} \tag{5}$$

We use equations 2 and 4 to derive the number of bins such that the signer has the probability of $P_S \approx 50\%$ to find a signature, given that the signer has t balls and the signature is composed of a k-way collision:

$$P_S \approx \frac{t^k}{k!n^{k-1}} = \frac{1}{2} \tag{6}$$

$$n^{k-1} = \frac{2t^k}{k!} \tag{7}$$

We can now compute the probability that the adversary can forge a signature, given that it has seen a single signature (hence it has k balls), and we use equation 7

$$P_f = \left(\frac{1}{n}\right)^{k-1} = \frac{k!}{2t^k} \tag{8}$$

For the two-round scheme, we assume that the signer has t balls, and that it looks for a k_1-way collision in the first round, and a k_2-way collision in the second round. It is clear that the signature comprises $k_1 \cdot k_2$ balls, so since we require that the signature size is the same for the one-round and two-round schemes we have $k = k_1 \cdot k_2$. We assume that the number n_1 of bins in the first round is fixed, and we compute the number of bins in the second round such that the probability of the signer P_S to find a signature is approximately 50%. The expected number of k_1-way collisions e_1 in the first round is

$$e_1 = \frac{t^{k_1}}{k_1! n_1^{k_1-1}} \tag{9}$$

We can now compute the number of bins n_2 of the second round

$$P_S \approx \frac{e_1^{k_2}}{k_2! n_2^{k_2-1}} = \frac{1}{2} \tag{10}$$

$$n_2^{k_2-1} = \frac{2e_1^{k_2}}{k_2!} = \frac{2t^{k_1 k_2}}{k_2!(k_1! n_1^{k_1-1})^{k_2}} \tag{11}$$

The probability that the attacker can find another signature with $k_1 \cdot k_2$ balls is

$$P_f = P_c \cdot \left(\frac{1}{n_2}\right)^{k_2-1} \tag{12}$$

where P_c is the probability to find k_2 k_1-way collisions after throwing $k_1 \cdot k_2$ balls into n_1 bins:

$$P_c = \frac{\binom{n_1}{k_2}(k_1 \cdot k_2)!}{(k_1!)^{k_2} n_1^{k_1 k_2}} \tag{13}$$

Combining equations 11, 12, and 13 we get

$$\begin{aligned} P_f &= \frac{\binom{n_1}{k_2}(k_1 \cdot k_2)!}{(k_1!)^{k_2} n_1^{k_1 k_2}} \frac{k_2!(k_1!)^{k_2} n_1^{(k_1-1)k_2}}{2t^{k_1 k_2}} \\ &= \frac{\binom{n_1}{k_2}(k_1 \cdot k_2)! k_2!}{(n_1)^{k_2} 2t^{k_1 k_2}} \\ &\approx \frac{(k_1 \cdot k_2)!}{2t^{k_1 k_2}} \end{aligned} \tag{14}$$

When we set $k_1 \cdot k_2 = k$ in Equation 14 we can see that the probability to forge a signature P_f is the same as in Equation 4, which shows that the two-round scheme offers no better security than a one-round scheme. □

THEOREM 1. *A one-round Biba signature is asymptotically as secure as a n-round Biba signature.*

The proof is in the full version of this paper.

B. MERKLE HASH TREES FOR SEAL AUTHENTICATION

We can use a Merkle hash tree for the SEAL authentication [12]. The signer chooses the SEALs randomly, places them at the leaves of a binary tree and computes each internal node as the hash of the concatenation of the two child values.[4] The root node of the hash tree becomes the public key, hence the public key is small.

The signer computes the signature as before, but it needs to add additional verification nodes of the hash tree to the signature, such that the verifier can reconstruct all the paths from all SEALs to the root of the hash tree. Unfortunately, the overhead of these additional verification nodes can be high, as we now compute.

We assume that the disclosed SEALs are distributed randomly in the hash tree. To compute the overhead, we need to compute the probability that all the leaves below a given tree node are all empty. We set function $\pi(a, r, n)$ as the probability that a leaves are empty after randomly choosing r leaves among n leaves (note that $\pi(a, r, n) = 0$ if $n - r < a$):

$$\pi(a, r, n) = \prod_{i=0}^{r-1} \frac{n-a-i}{n-i}$$

The expected number of nodes of the Merkle hash tree that need to be sent to authenticate b leaf nodes depends on the depth d of the hash tree. The number of expected nodes is:

$$\sum_{i=1}^{d} 2^i \pi(2^{d-i}, b, 2^d)(1 - \pi(2^{d-i}, r, 2^d - 2^{d-i}))$$

The intuition behind this formula is that we need to send a node of the tree if exactly one child has all empty leaf nodes, and the other child node has at least one chosen leaf node.

When we evaluate this probability for the schemes we list as examples 1 and 2, we would need to add 83.3 hash tree nodes on average to authenticate the 16 SEAL values for the scheme in the first example, and 67.5 nodes to authenticate the 12 SEALs in the second example. Clearly, this would increase the signature size by a factor of 6, and increase the verification overhead.

To generalize this idea, we can construct many small hash trees of height d that contain 2^d SEALs. The public key would then contain all the root nodes of all small hash trees, and hence we reduce the size by a factor of 2^d. To authenticate each SEAL, the signer adds the d verification nodes to each SEAL. Hence, the public key size is reduced by a factor of 2^d and the signature size is expanded by a factor of d.

[4] A minor point is that the signer needs to compute a one-way function on the SEAL before placing the SEAL at the leaf node. Otherwise the signer would disclose neighboring SEALs when it discloses additional nodes for verification.