

Efficient Multicast Stream Authentication for the Fully Adversarial Network Model*

Christophe Tartary and Huaxiong Wang

Centre for Advanced Computing, Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia

{ctartary, hwang}@ics.mq.edu.au

Abstract

We consider the stream authentication problem when an adversary has the ability to drop, reorder or inject data packets in the network. We propose a coding approach for multicast stream authentication using the list-decoding property of Reed-Solomon codes. We divide the data to be authenticated into a stream of packets and associate a single signature for every λn packets where λ and n are predesignated parameters. Our scheme, which is also joinable at the boundary of any n -packet block, can be viewed as an extension of Lysyanskaya, Tamassia and Triandopoulos's technique in which $\lambda = 1$. We show that by choosing λ and n appropriately, our scheme outperforms theirs in both signature and verification time. Our approach relies on signature dispersal as SAIDA and eSAIDA. Assuming that we use RSA for signing and MD5 for hashing, we give an approximation of the proportion of extra packets per block which could be processed via our technique with respect to the previous schemes. As example when we process $\lambda = 1000$ blocks of 20000 64-byte-packets, the gain of our scheme with respect to Lysyanskaya *et al.*'s is about 30%.

Keywords: Stream Authentication, Signature Dispersal, Reed-Solomon Codes.

1 Introduction

Broadcast communication enables a sender to distribute data to many receivers via a public communication channel such as the Internet. Their applications cover a large scope of areas such as software updates, sensor networks, GPS signals, pay-TV, stock quotes and military defense systems for instance. Nevertheless, existing IP protocols in the Internet only provide a best-effort delivery process and the large number of receivers prevents lost content from being redistributed. In addition malicious users having access to the network can perform harmful actions on the data stream. Thus, the security relies on two aspects: the network properties and opponents' computational power. In this paper, we will consider the computationally secure model for broadcast authentication. That is, the opponents have bounded computational abilities.

Many techniques have been designed to deal with multicast stream authentication [3]. Examples as pay-TV and stock quotes involve that data stream can be infinite and must be consumed as soon as they reach the receivers (or within a short delay). The most basic idea of signing each packet¹ is inappropriate, as digital signatures are typically time expensive. The available transmission bandwidth does not allow the use of one-time or k -time signatures [5, 21] either because of their large size whereas the construction of Boneh *et al.*'s short signatures [2] is too restrictive to be used in our case. Since signing each packet is prohibitive, other techniques rely on signature amortization. This means that one signature is produced and its cost (both in time and overhead) is amortized over several packets (due to hash functions for instance).

In [23], Wong and Lam built a Merkle-hash tree [10] to distribute hashes. Their scheme is tolerant against any kind of packet loss. Nevertheless, the tag² size is logarithmic in the number of packets per block.

In [5], Gennaro and Rohatgi proposed to sign the first stream packet and link the hash of each packet into the next one's tag. This approach needs the entire stream to be known in advance and if a single packet is lost then the whole process fails.

To deal with packet loss, Perrig *et al.* designed EMSS [17, 18] and MESS [18] by appending the hash of each packet to a fixed number of followers according to a specific pattern. One packet is signed from time to time to ensure non-repudiation and is always assumed to be received. They modeled the network loss pattern by a k -state Markov chain

*The original version of this paper appears in the proceedings of the 6th International Workshop on Information Security Applications (WISA 2005), Lecture Notes in Computer Science, vol. 3786, pp 108 - 125, Springer - Verlag, 2006.

¹Since the data stream is large it is divided into fixed-size entities called *packets*.

²We call *authentication tag* the extra information appended to a packet to provide its authenticity.

(see [16, 24]) and provided bounds on the packet verifiability. Considering the diversity of computational abilities within the set of receivers, Challal *et al.* [4] used different layers for hash distribution. Their H₂A protocol gave good practical improvements with respect to MESS. Golle and Modadugu [6] and Miner and Staddon [11] proved other bounds based on augmented chains. The main drawback of all these schemes is that they rely on the signature reception reliability (except Wong and Lam's one [23]). To overcome this problem, one possibility is to split the signature into k smaller parts where only ℓ of them ($\ell < k$) are enough for recovery.

The Information Dispersal Algorithm [20] has been used by Park *et al.* [13, 14] and Park and Cho [15] to design two similar schemes SAIDA and eSAIDA (the later having a better packet verification probability). Al-Ibrahim and Pieprzyk [1] used linear equations and polynomial interpolation whereas Pannetrat and Molva [12] proposed some erasure codes to achieve signature dispersal. Nevertheless, these four schemes share a common drawback: they do not tolerate a single packet injection.

Using an error-correcting code approach, Lysyanskaya *et al.* [8] designed a scheme resistant to packet loss and injections (provided some assumptions on the network delivery reliability). As the five previous schemes above, a single signature is created per block and amortized over several packets. These techniques extended the notion of packet signature to block signature. The scheme developed in [5] generates a single signature for the whole stream but does not tolerate a single packet loss.

In order to decrease time spent for signature generation and verification, our approach is to generate one signature for every family of λ blocks where each of them consists of n packets. The value of the parameter λ has to be chosen carefully by the sender since he will have to memorize λn packets at a time. Nevertheless, as in [8], data are sent and can be authenticated by receivers per block. This regulates the traffic in the network avoiding too irregular throughput variations which could create a bottleneck. The family signature is spread within each block which enables a receiver to join the communication group at any block boundary. The minimal value, Λ , of λ from which our protocol is faster than Lysyanskaya *et al.*'s one remains very small. For instance, we have $\Lambda = 2$ up to $n = 30000$ when using RSA and MD5 for 64-byte packets. This value for n is much larger than the one used by Perrig *et al.* to implement EMSS ($n = 1000$). The profit of our approach is significant. For instance, we have a benefit of at least 50% more packets per block with respect to Lysyanskaya *et al.*'s technique and linear equations' approach (up to $n = 11500$) and to SAIDA and eSAIDA (up to $n = 13300$). If $n = 1000$ (as for EMSS) then our technique provides a benefit larger than 90% more packets per block than Lysyanskaya *et al.*'s scheme.

The paper is organized as follows. In the following section, we describe the scheme developed in [8]. In Section 3, we will introduce our modifications and prove the security of this new scheme under similar assumptions to those made in [8]. In Section 4, we will compare our extended scheme to some above ones to get an idea of the gain it provides toward them. In Section 5, we will improve the signature verification complexity. The last section will summarize our contribution to the multicast stream authentication problem.

2 Preliminaries

Definition 1 A $[N, K]_q$ **systematic Reed-Solomon (SRS)** code over the finite field \mathbb{F}_q ($q > N$) is a function:

$$\mathcal{C} : \begin{array}{ccc} (\mathbb{F}_q \times \mathbb{F}_q)^K & \rightarrow & (\mathbb{F}_q \times \mathbb{F}_q)^N \\ \{(i, y_i)\}_{i \in \{1, \dots, K\}} & \mapsto & \{(i, p(i))\}_{i \in \{1, \dots, N\}} \end{array}$$

such that p is an element of $\mathbb{F}_q[X]$ of degree at most K with $\forall i \in \{1, \dots, K\} p(i) = y_i$. The rational $\frac{K}{N}$ (< 1) is called the rate of the code.

The code is called *systematic* since the first K symbols of any codeword are its corresponding message [9]. Given the K points $\{(i, y_i)\}_{i \in \{1, \dots, K\}}$, the polynomial p defined above is unique. In order to deal with the attack of packet injections, we will list-decode this SRS code using the Guruswami-Sudan decoder (GS-Decoder) developed in [7]. It is based on the polynomial reconstruction problem, takes as input integers K, t , and M couples of field elements $\{(\tilde{x}_i, \tilde{y}_i)\}_{i \in \{1, \dots, M\}}$, and outputs the list of all univariate polynomials \tilde{p} of degree at most K such that $\tilde{y}_i = \tilde{p}(\tilde{x}_i)$ for at least t values of $i \in \{1, \dots, M\}$. It is shown in [7] that if $t > \sqrt{KM}$ then the polynomial reconstruction problem could be solved in polynomial time. Then, it has been deduced that any $[N, K]_q$ Reed-Solomon code (systematic or not) with an error at most $N - t$ could be list-decoded using $O(N^2)$ field operations producing a list of $O(1)$ candidates.

We consider the scenario where the sender has much larger computational memory storage (to buffer a piece of the data stream) and computational abilities than the receivers. This illustrates most cases since, in general, the sender is a server delivering data to personal computers.

In the fully adversarial model, the adversary \mathcal{A} can introduce packets into the channel, drop and rearrange some chosen original ones. Thus, reliable transmission of the signature is not possible since \mathcal{A} would only need to drop the signature

packet to make the authentication scheme fail. Since the authentication problem is our major concern, we assume that a reasonable number of original packets reaches the receivers. Indeed if too many packets are discarded or modified by \mathcal{A} then the main problem becomes data transmission since the small number of packets reaching the receivers would be useless for their original purposes even authenticated. On the other hand if too many packets are received then prevention against denial-of-service attacks becomes the main concern. We split the stream into blocks of n packets and define two parameters:

- $\alpha (0 < \alpha \leq 1)$: the *survival* rate. At least αn original packets are received.
- $\beta (\beta \geq 1)$: the *flood* rate. A maximum of βn packets reaches each receiver.

We now briefly describe the scheme defined by Lysyanskaya *et al.* [8]. Let ρ be the rate of the SRS code we will use. Since $\frac{\alpha^2}{\beta} \in (0, 1]$, there exists $\epsilon > 0$ such that: $\rho = \frac{\alpha^2}{(1+\epsilon)\beta}$. ϵ is called the *tolerance parameter* of the decoder. The choice of ρ will be explained later. We use a signature scheme [22] (the key generator of which is Keygen) and a collision-resistant hash function [19]. Each block of n packets has an identification tag BID (representing its position within the whole stream). The authenticator Auth first hashes each packet and signs the concatenation of BID together with the n hashes. Then, we form the authentication stream S which is the concatenation of the n hashes and the signature. S is split into $\rho n + 1$ field elements over \mathbb{F}_q where $q = 2^{\lceil \frac{|S|}{\rho n + 1} \rceil}$ (after padding if necessary). S is encoded using the SRS $[n, \rho n]_q$ code giving n pieces of signature. Each authenticated packet is the concatenation of BID, the packet position within the block, the packet itself and the corresponding piece of signature.

From [7], we must have $t > \sqrt{KM}$ to ensure the success of GS-Decoder. In our case, we have $t = \alpha n$ (minimum number of original packets arrived at the receiver end), $K = \rho n$ and $M = m$ (number of received packets ($\alpha n \leq m \leq \beta n$)). Thus, from the inequality $t > \sqrt{KM}$, we have $\beta n > \frac{m}{1+\epsilon}$. So, GS-Decoder can be run successfully for our choice of ρ . Since ϵ has an impact on the success of that decoder, we denote it: **GS-Decoder $_\epsilon$** . To fit the fact that our code is systematic, we need to modify GS-Decoder $_\epsilon$ before using it for authenticating packets. The new algorithm **MGS-Decoder $_\epsilon$** is depicted as Algorithm 1.

Algorithm 1 MGS-Decoder $_\epsilon$

Input: The number of packets per block n , the network characteristics α, β and m elements $\{(x_i, y_i), 1 \leq i \leq m\}$.

1. If $(m > \beta n)$ or (we have less than αn distinct values of x_i) then the algorithm rejects the input.
2. Run GS-Decoder $_\epsilon$ on the m elements to get a list L of polynomials. Evaluate each $Q_i(X)$ at $1, \dots, \rho n + 1$ and concatenate these values to form c_i .

Output: $\{c_1, \dots, c_{|L|}\}$: list of candidates.

We notice that since α, β and ϵ are known, ρ can be easily computed. Thus, there is no need to consider it as an input. Now, we describe the decoding algorithm *Decoder $_\epsilon$* used in [8]. After verifying that the number of packets with suitable BID and packet numbering is between αn and βn , MGS-Decoder $_\epsilon$ is run to obtain a list of candidates for signature verification. The list is processed until the signature is checked or the whole list is exhausted. If the MGS-Decoder $_\epsilon$ rejects the input or the list is processed in vain then the family of received packets is dropped. Otherwise (i.e. the signature has been verified successfully), the good candidate is split as above (as the concatenation of the BID and n hashes). Then, each of the received packets is processed and we check whether its hash matches one of the n ones. If so, the corresponding packet is output as authentic. We now describe the improvements we made on Lysyanskaya *et al.*'s scheme.

3 Our Protocol

Our work is an extension of the scheme described in Section 2 [8]. Since signatures are time expensive to generate and verify, our idea is to compute one signature for a family of λ blocks where each block consists of n packets. We assume that the sender can buffer λn packets. Nevertheless, our scheme works in such a way that a receiver only needs to get enough packets from a block before verifying it (he does not have to wait for the whole sequence). As the previous scheme, it will be joinable at any block boundary. We need a collision-resistant hash function h as well as a signature scheme (KeyGen, sign, verify) where KeyGen generates the private key SK and its corresponding public key PK. We denote $\|\cdot\|$ the concatenation of two elements. Figure 1 gives a description of the sender's work for the sequence of blocks $\{B_1, \dots, B_\lambda\}$.

We keep the same definitions for n, α, β, ρ and ϵ as before. Each family $\{P_1^1, \dots, P_\lambda^n\}$ of λn packets of the stream has an identification tag FID representing its position within the whole stream. Each one of its blocks of n packets also has a tag BID. Thus, a packet is now identified within the stream by its position i within a block BID belonging to the family FID, i.e. its *identification number* is (FID, BID, i). We now describe the family authenticator AuthFamily which outputs the packets per block of n elements.

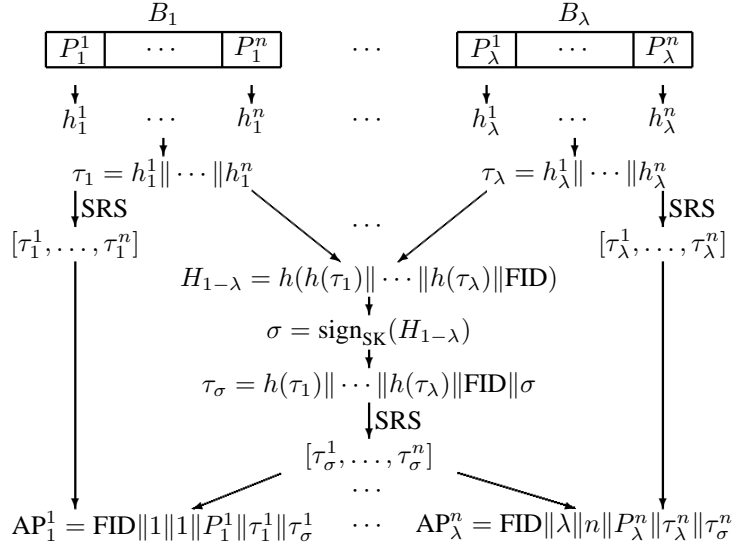


Figure 1: Authentication process of the extended scheme.

Algorithm 2 AuthFamily

Input: The private key SK , the network characteristics α, β , a family $\{P_1^1, \dots, P_\lambda^n\}$, its FID and λ .

1. Within each block b we hash the n packets and concatenate them to form the block tag τ_b . It is then encoded using the SRS code and we get $(\tau_b^1, \dots, \tau_b^n)$.
2. Hash the λ block tags and concatenate them together with FID. This concatenation is hashed to get $H_{1-\lambda}$ which is signed to form σ using SK . The resulting signature is encoded using the SRS code and we get $(\tau_\sigma^1, \dots, \tau_\sigma^n)$.
3. The λn authenticated packets are defined as $\text{AP}_b^p = \text{FID} \parallel b \parallel p \parallel P_b^p \parallel \tau_b^p \parallel \tau_\sigma^p$. As soon as the n packets of a block are processed then the whole block is sent immediately.

Output: $\{\text{AP}_1^1, \dots, \text{AP}_\lambda^n\}$: set of authenticated packets sent per block of n packets.

In order to use the same SRS code, τ_σ and the τ_b 's ($1 \leq b \leq \lambda$) must be padded appropriately. If we denote \mathcal{H} the size of a hash, s the signature size and $|\cdot|$ the mapping giving the size of an element then $|\tau_b| = n\mathcal{H}$ and $|\tau_\sigma| = \lambda\mathcal{H} + s + |\text{FID}|$. In our work, we can assume that $\lambda < n$. Otherwise, our scheme requires the sender to buffer too many packets to preserve the live diffusion of data. We can also assume that $|\text{FID}|$ does not exceed the size of a hash. Thus, we can assume that $|\tau_b| < n\mathcal{H} + s$ and $|\tau_\sigma| < n\mathcal{H} + s$. So, we will use in both cases the $\text{SRS}[n, \rho n]_q$ code where q is the same integer as in Sect. 2. Thus, our extension does not increase the size of the field we work with. τ_σ and τ_i 's are padded according to that finite field.

If we do not take into account the identification number then any packet's tag is $\tau_{\text{BID}}^i \parallel \tau_\sigma^i$ which is the concatenation of 2 field elements. Once a generator of the extension $\mathbb{F}_q/\mathbb{F}_2$ is chosen then any element of \mathbb{F}_q requires $\log_2(q)$ bits. So, our tag is as large as $2 \log_2(q)$ bits which is approximately $\frac{2}{\rho}\mathcal{H}$ bits. Since $\rho < 1$, the tag is slightly larger than two hashes produced by h .

Since each block carries the signature, it is sufficient to run the signature verification process for family FID until one of its blocks makes the authentication process successful. Therefore, when a new block of packets is received, the receiver must react differently whether the family signature has already been verified or not. We first design the signature verification routine `VerifySignatureFamily`.

Algorithm 3 VerifySignatureFamily

Input: The public key PK , the network characteristics α, β , a set of pairs of field elements $\{(x_i, y_i), 1 \leq i \leq m\}$, the family FID and λ .

1. Run $\text{MGS-Decoder}_\epsilon$ on $\{(x_i, y_i), 1 \leq i \leq m\}$ to get a list L of candidates for the family signature verification. If $\text{MGS-Decoder}_\epsilon$ rejects this input then the algorithm stops.
2. While the signature has not been verified and the list L has not been exhausted, we pick a new candidate $\tilde{h}(\tau_1) \parallel \dots \parallel \tilde{h}(\tau_\lambda) \parallel \tilde{\sigma}$. If $\text{verify}_{\text{PK}}(h(\tilde{h}(\tau_1) \parallel \dots \parallel \tilde{h}(\tau_\lambda) \parallel \text{FID}), \tilde{\sigma}) = \text{TRUE}$ then $\tilde{\sigma}$ is considered as the authentic family signature σ and the $\tilde{h}(\tau_i)$'s are memorized within the table `HashBlock` as the authentic hash blocks $h(\tau_i)$'s.
3. If the signature has not been verified then our algorithm stops.

Output: $(\sigma, \text{HashBlock})$: family signature and hashes of the λ blocks.

Now, we describe our block decoder $\text{DecoderBlock}_\epsilon$. The definition of the boolean TestSignature is necessary because our scheme only checks the family signature until it is verified by one block within the family FID. Once this has been done, block hashes are stored into HashBlock and only block authentications are performed. Let $\text{RP} = \{R_1, \dots, R_m\}$ be the set of received packets.

Algorithm 4 $\text{DecoderBlock}_\epsilon$

Input: The public key PK , the network characteristics $\alpha, \beta, n, \text{FID}, \text{BID}, \lambda$, a boolean TestSignature , a table HashBlock and the set of received packets RP .

1. Write the packets as $\text{FID}_i \parallel \text{BID}_i \parallel j_i \parallel P_{\text{BID}_i}^{j_i} \parallel \tau_{\text{BID}_i}^{j_i} \parallel \tau_\sigma^{j_i}$ and discard those having $\text{FID}_i \neq \text{FID}, \text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \dots, n\}$. Denote m' the number of remaining packets. If $m' < \alpha n$ or $m' > \beta n$ then the algorithm stops.
2. If $(\text{TestSignature} = \text{TRUE})$ then go to step 3. Otherwise, run $\text{VerifySignatureFamily}$ on the m' remaining points. If it rejects the input then the algorithm stops. Otherwise, set $\text{TestSignature} = \text{TRUE}$.
3. Run $\text{MGS-Decoder}_\epsilon$ on the set $\{(j_i, \tau_\sigma^{j_i}), 1 \leq i \leq m'\}$ and get a list L of candidates for block tag verification. If $\text{MGS-Decoder}_\epsilon$ rejects that set then the algorithm stops.
4. While the block BID has not been verified and the list L has not been exhausted, we pick a new candidate $\tilde{c} := \tilde{h}_{\text{BID}}^1 \parallel \dots \parallel \tilde{h}_{\text{BID}}^n$. If $(h(\tilde{c}) = \text{HashBlock}(\text{BID}))$ then the tag of block BID is verified and we set $h_{\text{BID}}^j = \tilde{h}_{\text{BID}}^j$ for $j \in \{1, \dots, n\}$. If L is exhausted without a successful block tag verification then the algorithm stops.
5. For $i \in \{1, \dots, n\}$, set $P_{\text{BID}}^i = \emptyset$. For each packet of RP (written as $R_{\text{BID}}^i = \text{FID} \parallel \text{BID} \parallel j \parallel P_{\text{BID}}^j \parallel \tau_{\text{BID}}^j \parallel \tau_\sigma^j$ where $j \in \{1, \dots, n\}$) if $h(P_{\text{BID}}^j) = h_{\text{BID}}^j$ then $P_{\text{BID}}^j = P_{\text{BID}}^j$.

Output: $\{P_{\text{BID}}^1, \dots, P_{\text{BID}}^n\}$: set of identified packets.

After Step 1, the remaining m' packets are renumbered as $\{R_{\text{BID}}^1, \dots, R_{\text{BID}}^{m'}\}$ where $R_{\text{BID}}^i = \text{FID} \parallel \text{BID} \parallel j_i \parallel P_{\text{BID}}^{j_i} \parallel \tau_{\text{BID}}^{j_i} \parallel \tau_\sigma^{j_i}$. When we enter Step 4, the table HashBlock is full since the family signature has been verified.

Since a single signature is created per family of λ blocks, one might think that our scheme is only joinable at a family boundary. Nevertheless, $[\tau_\sigma^1, \dots, \tau_\sigma^n]$ is present within each block of n packets the sender emits. Thus, any receiver can join the communication group at any block boundary as in [8].

Since the families of λ blocks are independent from each other, the security of our scheme relies on the security of a family of λ blocks. Similar to [8], we give the following definition.

Definition 2 (*KeyGenerator, Authenticator, Decoder*) is a **secure** and **(α, β) -correct** multicast authentication scheme if no probabilistic polynomial-time adversary \mathcal{A} can win with a non-negligible probability to the following game:

- (i) A key pair (SK, PK) is generated by *KeyGenerator*.
- (ii) \mathcal{A} is given: (a) The public key PK and (b) Oracle access to *Authenticator* (but \mathcal{A} can only issue at most one query with the same family identification tag FID).
- (iii) \mathcal{A} outputs $(\text{FID}, n, \alpha, \beta, \text{RP})$.

\mathcal{A} wins if one of the following happens:

- (a) (violation of the correctness property) The adversary succeeds to output RP such that even if it contains $\alpha_i n_i$ packets of some authenticated packet set AP_i for family identification tag $\text{FID}_i = \text{FID}$ and block identification tag $\text{BID}_i = \text{BID}$, the decoder still fails at authenticating some of the correct packets.
- (b) (violation of the security property) The adversary succeeds to output RP such that the decoder outputs $\{P_{\text{BID}}^1, \dots, P_{\text{BID}}^n\}$ (for some BID) that were never authenticated by *Authenticator* (as a part of a family of λ blocks) for the family tag FID .

Lysyanskaya *et al.* [8] showed that their scheme (*Keygen, Auth, Decoder* $_\epsilon$) was secure and (α, β) -correct. Following their arguments, we obtain the following result for our scheme.

Theorem 1 The authentication scheme (*KeyGen, AuthFamily, DecoderBlock* $_\epsilon$) is secure and (α, β) -correct.

Proof.

Suppose that our scheme is neither secure nor (α, β) -correct. By definition, an adversary \mathcal{A} can break the scheme with a non-negligible probability $\mathcal{P}(k)$. We have:

$$\begin{aligned}
\mathcal{P}(k) &= p(\{\text{the scheme is not } (\alpha, \beta)\text{-correct or insecure}\}) \\
&= p(\{\text{the scheme is not } (\alpha, \beta)\text{-correct}\} \cup \{\text{the scheme is insecure}\})
\end{aligned}$$

Since p is a measure, we deduce that one of the following two cases is true:

$$p(\{\text{the scheme is not } (\alpha, \beta)\text{-correct}\}) \geq \frac{\mathcal{P}(k)}{2} \quad (1)$$

$$p(\{\text{the scheme is insecure}\}) \geq \frac{\mathcal{P}(k)}{2} \quad (2)$$

Point (1). If a polynomial-time adversary \mathcal{A} breaks the (α, β) -correctness of the scheme then the digital signature scheme can be forged. This will be proved by turning an attack breaking the (α, β) -correctness into an attack against the signature scheme. For this attack, \mathcal{A} has access to the signing algorithm sign_{SK} (but not SK itself), can use the public signature key PK and the cryptographic hash function h . He is also able to run the authentication scheme AuthFamily. The queries made to it are written as $(\text{FID}_i, \lambda_i, n_i, \alpha_i, \beta_i, DP_i)$ where DP_i is the set of $\lambda_i n_i$ data packets to be authenticated. In order to get the corresponding output, the signature is obtained by querying sign_{SK} within the authenticator. Following this process, \mathcal{A} is able to break the scheme correctness since he got values FID, $\lambda, n, \alpha, \beta$ and a set of received packets RP_{BID} (for some $\text{BID} \in \{1, \dots, \lambda\}$) such that:

- $\exists i / (\text{FID}, \lambda, n, \alpha, \beta) = (\text{FID}_i, \lambda_i, n_i, \alpha_i, \beta_i)$
Denote $DP = \{P_1^1, \dots, P_\lambda^n\} (= DP_i)$ the data packets associated with this query and AP the response given to \mathcal{A} . In particular, we denote $\sigma = \text{sign}_{\text{SK}}(H_{1-\lambda})$ where $H_{1-\lambda} = h(h(\tau_1) \parallel \dots \parallel h(\tau_\lambda) \parallel \text{FID})$ with $\forall j \in \{1, \dots, \lambda\} \tau_j = h(P_j^1) \parallel \dots \parallel h(P_j^n)$.
- $|RP_{\text{BID}} \cap AP| \geq n\alpha$ and $|RP_{\text{BID}}| \leq \beta n$.
- $(P_{\text{BID}}^1, \dots, P_{\text{BID}}^n) = \text{DecoderBlock}_\epsilon(\text{PK}, \text{FID}, \text{BID}, n, \alpha, \beta, \text{TestSignature}, \text{HashBlock}, RP_{\text{BID}})$ where for some j such that $R_{\text{BID}}^j \in RP$ we have $P_{\text{BID}}^j \neq P_{\text{BID}}^j$ with: $R_{\text{BID}}^j = \text{FID} \parallel \text{BID} \parallel j \parallel P_{\text{BID}}^j \parallel \tau_{\text{BID}}^j \parallel \tau_\sigma^j$.

Since $\text{DecoderBlock}_\epsilon$ first checks the family signature and second outputs packets, TestSignature can take two different values (each of them involves a specific value of HashBlock). Thus, \mathcal{A} must be able to succeed in both following cases:

- The set RP_{BID} is used to verify the signature.
- The signature of the family has already been checked.

Case B illustrates the event when the receiver has already verified the family signature when he receives fake packets introduced by \mathcal{A} .

Case A. Since the set RP_{BID} verifies the signature, the query to $\text{DecoderBlock}_\epsilon$ gives us a candidate $c' = h'_1 \parallel \dots \parallel h'_\lambda \parallel \sigma'$ with $\text{verify}_{\text{PK}}(h(h'_1 \parallel \dots \parallel h'_\lambda \parallel \text{FID}), \sigma') = \text{TRUE}$. We have to prove that sign_{SK} was not run on the input $h(h'_1 \parallel \dots \parallel h'_\lambda \parallel \text{FID})$. This is proved in [8]. Here, we have a slight difference. That is, we have $h(h'_1 \parallel \dots \parallel h'_\lambda \parallel \text{FID})$ whereas [8] deals with $h'_1 \parallel \dots \parallel h'_\lambda \parallel \text{FID}$. As h is collision resistant, this difference is not a problem.

Case B. Now, we consider that the signature has previously been verified. That is, the receiver has buffered h'_1, \dots, h'_λ and σ' such that $\text{verify}_{\text{PK}}(h(h'_1 \parallel \dots \parallel h'_\lambda \parallel \text{FID}), \sigma') = \text{TRUE}$. We have two possibilities: $P_{\text{BID}}^j \neq \emptyset$ or $P_{\text{BID}}^j = \emptyset$.

- Sub-case B1: $P_{\text{BID}}^j \neq \emptyset$. Since h is collision-resistant, we have $h(P_{\text{BID}}^j) \neq h(P_{\text{BID}}^j)$. Since P_{BID}^j is a non-empty part of a received packet, the decoding algorithm $\text{DecoderBlock}_\epsilon$ outputs a candidate $c'_{\text{BID}} = h'^1_{\text{BID}} \parallel \dots \parallel h'^n_{\text{BID}}$ such that $h'_{\text{BID}} = h(h'^1_{\text{BID}} \parallel \dots \parallel h'^n_{\text{BID}})$. Moreover, $\text{DecoderBlock}_\epsilon$ includes P_{BID}^j into the output packets if and only if $h(P_{\text{BID}}^j) = h'^j_{\text{BID}}$. Remember that $h(P_{\text{BID}}^j) \neq h(P_{\text{BID}}^j)$. We get:

$$h(P_{\text{BID}}^1) \parallel \dots \parallel h(P_{\text{BID}}^j) \parallel \dots \parallel h(P_{\text{BID}}^n) \neq h(P_{\text{BID}}^1) \parallel \dots \parallel h(P_{\text{BID}}^j) \parallel \dots \parallel h(P_{\text{BID}}^n)$$

Since h is a collision-resistant, we get: $h'_{\text{BID}} \neq h_{\text{BID}}$ and for the same reason:

$$h(h(P_{\text{BID}}^1) \parallel \dots \parallel h(P_{\text{BID}}^j) \parallel \dots \parallel h(P_{\text{BID}}^n)) \neq h(h(P_{\text{BID}}^1) \parallel \dots \parallel h(P_{\text{BID}}^j) \parallel \dots \parallel h(P_{\text{BID}}^n))$$

Thus, the digital signature is not secure.

- Sub-case B2: $P_{\text{BID}}^j = \emptyset$. Due to the consistency of $\text{MGS-Decoder}_\epsilon$ (see [8]), $\text{DecoderBlock}_\epsilon$ will include the candidate value $c = h^1_{\text{BID}} \parallel \dots \parallel h^n_{\text{BID}}$. By definition, we have: $h(c) = h_{\text{BID}}$. So, the decoder cannot provide $(h^1_{\text{BID}}, \dots, h^n_{\text{BID}}) = (\emptyset, \dots, \emptyset)$. If $(h^1_{\text{BID}}, \dots, h^n_{\text{BID}}) = (h^1_{\text{BID}}, \dots, h^n_{\text{BID}})$ then the design of $\text{DecoderBlock}_\epsilon$ involves that $P_{\text{BID}}^j = P_{\text{BID}}^j$ be non-empty and R_{BID}^j is a received packet. In order to avoid this contraction, we must have $(h^1_{\text{BID}}, \dots, h^n_{\text{BID}}) \neq (h^1_{\text{BID}}, \dots, h^n_{\text{BID}})$. Nevertheless, $h(h^1_{\text{BID}} \parallel \dots \parallel h^n_{\text{BID}}) = h_{\text{BID}}$ which is impossible since h is collision-resistant. Thus, we

get a contradiction.

Point (2). If a polynomial-time adversary \mathcal{A} breaks the security property of the scheme than the underlying signature scheme is not secure. We consider the same kind of scheme as in Point (1). \mathcal{A} will succeed if one of the following will hold:

- A. AuthFamily was never queried on input $\text{FID}, \lambda, n, \alpha, \beta, DP$ and the decoding algorithm $\text{DecoderBlock}_\epsilon$ does not reject it, i.e. $OP_{\text{BID}} \neq \emptyset$ where $OP_{\text{BID}} = \text{DecoderBlock}_\epsilon(\text{PK}, \text{FID}, \text{BID}, n, \alpha, \beta, RP_{\text{BID}})$ with $\text{BID} \in \{1, \dots, \lambda\}$.
- B. AuthFamily was queried on input $\text{FID}, \lambda, n, \alpha, \beta, DP$. However some non-empty output packet P'_{BID}^j is different from P_{BID}^j where $OP_{\text{BID}} = \{P_{\text{BID}}^1, \dots, P_{\text{BID}}^n\}$ and $DP = \{P_1^1, \dots, P_\lambda^n\}$ for some $\text{BID} \in \{1, \dots, \lambda\}$.

Case A. Due to the design of $\text{DecoderBlock}_\epsilon$, the only possibilities to output non-empty packets were either (exhibiting a valid signature and valid hashes for block BID) or (valid hashes for block BID which are consistent with the signature and block hashes already buffered).

- Sub-case A1. Because we exhibit a valid signature, the $\text{MGS-Decoder}_\epsilon$ has output an element $c' = h'(\tau_1) \parallel \dots \parallel h'(\tau_\lambda) \parallel \sigma'$ such that $\text{verify}_{\text{PK}}(h(h'(\tau_1) \parallel \dots \parallel h'(\tau_\lambda) \parallel \text{FID}), \sigma') = \text{TRUE}$. Since AuthFamily was never queried with FID , neither does $\text{verify}_{\text{PK}}$. Thus, σ' is a successful forgery, that is the signature scheme is not secure.

- Sub-case A2. Denote σ and $h(\tau_1), \dots, h(\tau_\lambda)$ the valid signature and its corresponding tags of blocks. Since $P'_{\text{BID}}^j \neq P_{\text{BID}}^j$ we deduce: $h(P'_{\text{BID}}^j) \neq h(P_{\text{BID}}^j)$ because h is collision-resistant. For the same reason $h(\tau'_{\text{BID}}) \neq h(\tau_{\text{BID}})$. We get a contradiction since outputting packets involves $h(\tau'_{\text{BID}}) = h(\tau_{\text{BID}})$. We notice that even if we do not know all P_{BID}^i 's (some can be empty), the hash $h(\tau'_{\text{BID}})$ is known thanks to $\text{VerifySignatureFamily}$.

Case B. Here, we have the same situation as Point (1) Case A and Sub-case B2. We get a contradiction with the security of the signature scheme. □

Thus, our modifications do not weaken either the security or the correctness of the technique developed in [8]. In order to compare our protocol to those relying on the same principle, namely signature dispersal, we need to compute its cost.

AuthFamily requires $\lambda(n+1) + 1$ hashes, 1 signature generation and, based on the analysis of Lysyanskaya *et al.*'s scheme, $O(\lambda n \log n)$ field operations over \mathbb{F}_q .

$\text{DecoderBlock}_\epsilon$ is more complex to analyze since its complexity depends on the block used to successfully verify the family signature. First, we compute the cost generated by one block, say b ($1 \leq b \leq \lambda$), assuming that we received k packets where $k \leq \beta n$ and at least αn with right numbering. In the following, the field is the one used for the SRS code. We have two cases:

1. *The signature of the family has not been verified yet.* $\text{MGS-Decoder}_\epsilon$ is run in $O(n^2)$ field operations and outputs a list of $O(1)$ signature candidates. We compute one hash and one signature verification for each of them until the signature be verified. So, there is a total of $O(n^2)$ field operations, $O(1)$ hashes and $O(1)$ signature verifications.
2. *The signature of the family has already been verified.* $\text{MGS-Decoder}_\epsilon$ is run as above. Each element of the list is hashed. Then, $O(k)$ hashes are computed to authenticate the packets. Since $k \leq \beta n$ and β is constant, we have $k = O(n)$. So, there is a total of $O(n^2)$ field operations and $O(n)$ hashes.

Consider the whole family of λ blocks. We notice that block authentications are only processed after a successful signature verification. Denote \mathcal{B} the block which verifies the family signature. From block 1 to $\mathcal{B} - 1$, only unsuccessful signature verifications are performed (Case 1). For block \mathcal{B} , one successful verification and one block authentication are performed (both cases). For block $\mathcal{B} + 1$ to λ , only block authentications are performed (Case 2). We deduce the cost of the group of λ blocks:

$$\begin{aligned} & O((\mathcal{B} - 1)n^2 + n^2 + (\lambda - (\mathcal{B} + 1) + 1)n^2) \text{ field operations} \\ & O(\mathcal{B} - 1 + n + 1 + (\lambda - (\mathcal{B} + 1) + 1)n) \text{ hashes} \\ & O(\mathcal{B} - 1 + 1) \text{ signature verifications} \end{aligned}$$

So, we have $O(\lambda n^2)$ field operations, $O(\mathcal{B} + (\lambda - \mathcal{B})n)$ hashes and $O(\mathcal{B})$ signature verifications. We notice that the field operations complexity does not depend on the block \mathcal{B} . We also have $\mathcal{B} \leq \lambda$, so $\mathcal{B} = O(\lambda)$. Therefore, we have $O(\lambda n)$ hashes and $O(\lambda)$ signature verifications. Nevertheless, this kind of approximation is not relevant since the number of hashes depends on the signature verifications performed. Assuming $\mathcal{B} = O(\lambda)$, we lose this dependence and therefore get two "upper bounds" which are not reached at the same time.

4 Comparison of Signature Dispersal-based Schemes

4.1 Complexity Comparison

Our scheme relies on signature dispersal so we will compare it to SAIDA, eSAIDA, linear equations scheme and the Lysyanskaya *et al.*'s one. We will not consider erasure codes from [12] since they do not specify a particular class of codes. Thus, we cannot evaluate the complexity of this technique. The results of Table 1 are built based on the definitions found in [13, 14, 15, 1, 8] where SAIDA, e-SAIDA, linear equations scheme and Lysyanskaya *et al.*'s scheme are iterated λ times.

	Sender			Receiver		
	Field Op.	Hash	Signature	Field Op.	Hash	Sign. Verif.
SAIDA	$O(\lambda n^2)$	$\lambda(n+1)$	λ	$O(\lambda n^2)$	$O(\lambda n)$	λ
e-SAIDA	$O(\lambda n^2)$	$\lambda(\frac{3n}{2}+1)$	λ	$O(\lambda n^2)$	$O(\lambda n)$	λ
Linear Equations	$O(\lambda n^3)$	$\lambda(n+1)$	λ	$O(\lambda n^2)$	$O(\lambda n)$	λ
Lysyanskaya <i>et al.</i> 's Scheme	$O(\lambda n \log n)$	λn	λ	$O(\lambda n^2)$	$O(\lambda n)$	$O(\lambda)$
Our Scheme	$O(\lambda n \log n)$	$\lambda n + \lambda + 1$	1	$O(\lambda n^2)$	$O(\lambda n)$	$O(\lambda)$

Table 1: Cost for signature dispersal-based schemes.

We notice that the approach of [8] is much more efficient than the other three schemes on every category but signature verification. Nevertheless, this is where its strength against packet loss is. So, we can say that it is the most efficient technique using signature dispersal (amongst those quoted above). So, our focus is to compare it (when iterated λ times) to our technique. At the receiver, the complexities of both schemes seem to be equivalent but bounds (for our work) concerning hashes and signature verifications are linked together and their exact values are smaller (see Section 3). So, the complexity at the receiver is slightly better for our scheme. In Section 5, we will define a property for the rates α and β allowing $O(1)$ for signature verifications. At the sender, we experiment the same field operations complexity but our technique computes a single signature whereas the other scheme generates λ of them. This is at the cost of $\lambda + 1$ more hashes computations. As said before, generating a digital signature is more time expensive than computing a hash. Since a hash function takes inputs of any length, the time spent hashing the extra quantity generated by our scheme will be more relevant than the number of extra hashes itself to get an approximation of the gain provided.

4.2 Threshold Values

Denote \mathcal{H} the size of a hash (in bytes), t_h is time needed to hash one byte and t_s the time needed to produce one signature (both t_h and t_s must be expressed in the same unity). The extra $(\lambda + 1)$ hashes are $h(\tau_1), \dots, h(\tau_\lambda)$ and $H_{1-\lambda}$. We have:

$$\forall i \in \{1, \dots, \lambda\} |\tau_i| = n\mathcal{H} \quad \text{and} \quad |h(\tau_1)| \dots |h(\tau_\lambda)| \text{FID} = \lambda\mathcal{H} + |\text{FID}|$$

If we assume that $|\text{FID}|$ is negligible with respect to \mathcal{H} then the size of the extra quantity to be hashed is $(n+1)\lambda\mathcal{H}$. Since our scheme experiences $(\lambda - 1)$ less signatures we deduce that it is the faster one if and only if:

$$(n+1)\lambda\mathcal{H}t_h < (\lambda-1)t_s \iff \left(1 - (n+1)\mathcal{H}\frac{t_h}{t_s}\right)\lambda > 1 \quad (3)$$

Denote $\mathcal{K} := 1 - (n+1)\mathcal{H}\frac{t_h}{t_s}$. If $\mathcal{K} < 0$ then $\lambda < 1/\mathcal{K}$. This upper bound is logical. Indeed, $\mathcal{K} < 0$ means $t_s < (n+1)\mathcal{H}t_h$. Since t_h is small (in comparison to t_s) and \mathcal{H} not too large, this configuration happens when n is large enough. In that case, we have a lot more hashes per block. Thus, if λ is too large then it is faster to compute one signature per block than all the extra hashes plus the family signature. We are interested in the case where n is reasonable and so $\mathcal{K} > 0$. Since $\lambda > \frac{1}{\mathcal{K}}$, we define $\Lambda := \lceil \frac{1}{\mathcal{K}} \rceil$ (which depends on n). We implemented the mapping $n \mapsto \Lambda(n)$ with different hash functions (MD5, SHA-1, SHA-256, RIPEMD-160 and Panama Hash (little and big endian)) and signature schemes (RSA, DSA (both produce a 1024-bit signature) and ESIGN (1023 bits)). The graphs are depicted as Figure 2.

When Perrig *et al.* implemented EMSS [17, 18], one signature packets was sent every 1000 ones. Park and Cho [15] used $n = 200$ and $n = 512$ to implement both SAIDA and eSAIDA. Figure 2 shows that $\Lambda = 2$ when n is up to 1000 for our choice of hash functions and signature schemes.

Once Λ has been chosen as on Figure 2, we determine the gain in term of proportion of extra packets per block our scheme provides. That is, once $\lambda \geq \Lambda$ and n are fixed (Condition (3) being checked) we determine $\frac{n-\tilde{n}}{n}$ where \tilde{n} is defined such that processing a family of $\lambda\tilde{n}$ packets with our technique is as time consuming as λ consecutive iterations of Lysyanskaya *et al.*'s one with n packets per block. We also want to compute the gain of our model with respect to the schemes previously quoted. As before, we need to determine the time spent at the sender for both schemes ($|\text{FID}|$ will be considered as negligible). We denote \mathcal{P} the size of a packet (in bytes). Results are shown in Table 2.

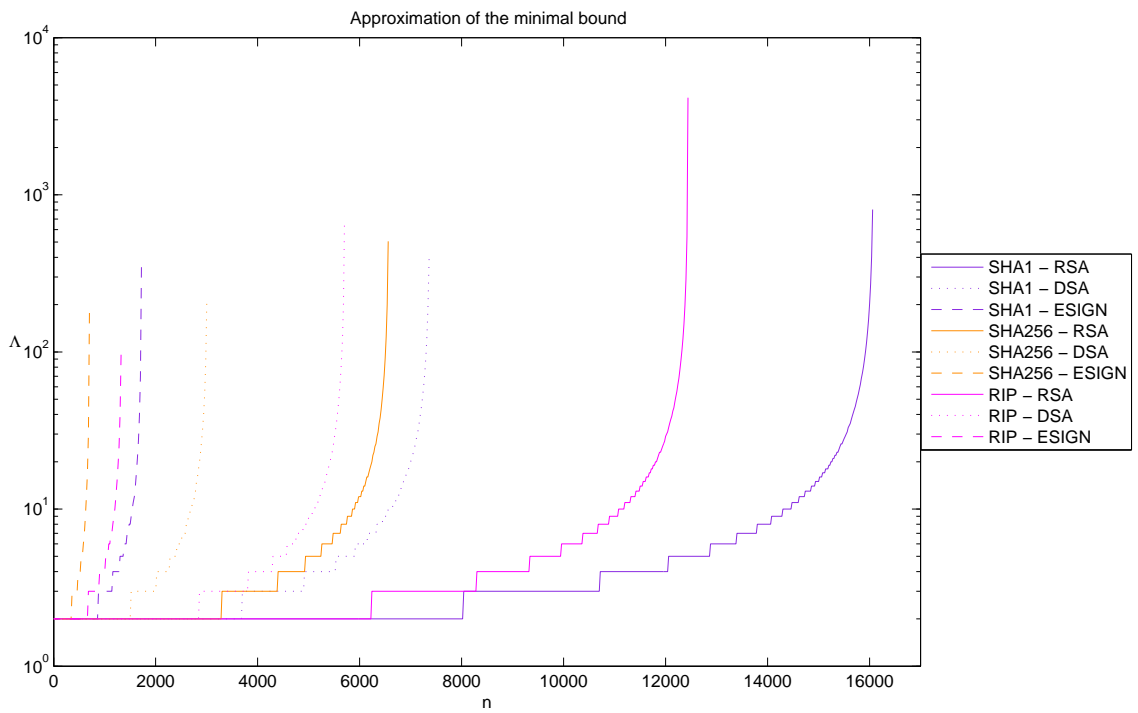
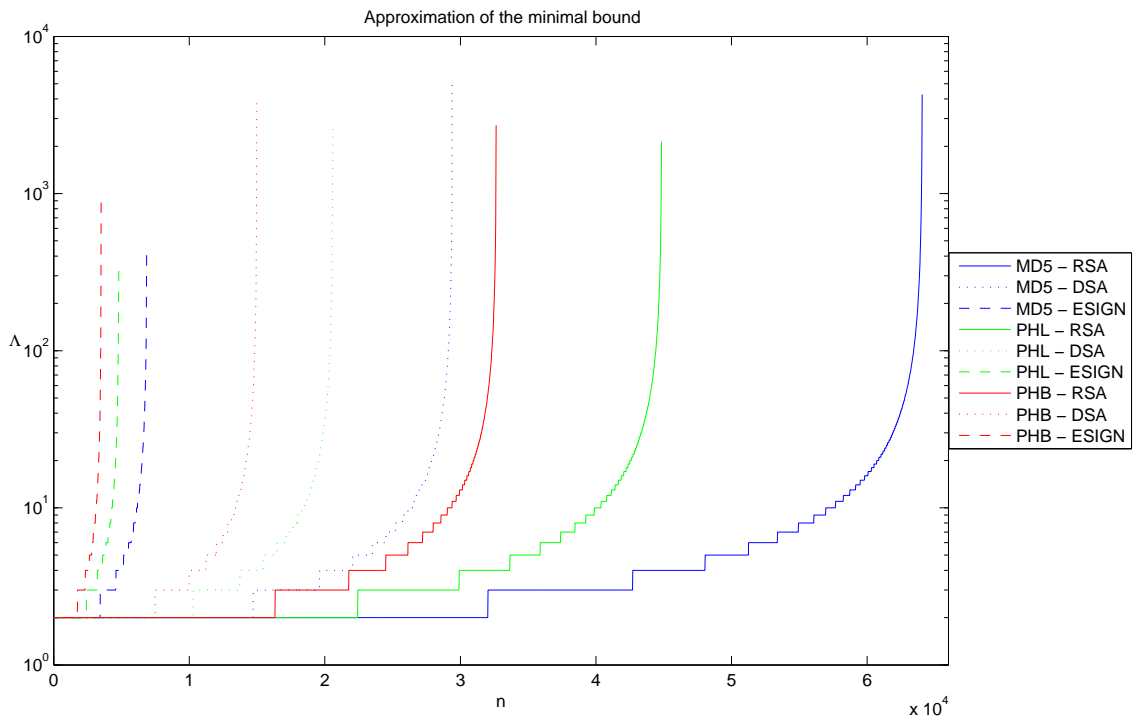


Figure 2: Computations of Λ for our different signature schemes and hash functions.

Our Extension	Lysyanskaya <i>et al.</i> 's Scheme (λ times)	Linear Equations (λ times)	SAIDA (λ times)	e-SAIDA (λ times)
$\lambda[n(\mathcal{P} + \mathcal{H}) + \mathcal{H}]t_h + t_s$	$\lambda(n\mathcal{P}t_h + t_s)$		$\lambda[n(\mathcal{P} + \mathcal{H})t_h + t_s]$	

Table 2: Time at the sender.

Thus, it is sufficient to study Lysyanskaya *et al.*'s scheme and SAIDA. Denote $T_{\text{EX}}^{\lambda,n}, T_{\text{LY}}^{\lambda,n}, T_{\text{SAIDA}}^{\lambda,n}$ the time spent at the sender for our protocol, Lysyanskaya *et al.*'s technique and SAIDA respectively with λ blocks of n packets each. We want to determine the minimal integers \mathcal{N}_{LY} and $\mathcal{N}_{\text{SAIDA}}$ such that:

$$\forall \mathcal{N} \geq \mathcal{N}_{\text{LY}} \quad T_{\text{EX}}^{\lambda,\mathcal{N}} > T_{\text{LY}}^{\lambda,n} \quad \text{and} \quad \forall \mathcal{N} \geq \mathcal{N}_{\text{SAIDA}} \quad T_{\text{EX}}^{\lambda,\mathcal{N}} > T_{\text{SAIDA}}^{\lambda,n}$$

This is equivalent to:

$$\mathcal{N}_{\text{LY}} = \left\lceil \frac{\mathcal{P}}{\mathcal{P} + \mathcal{H}} n + \frac{(1 - \frac{1}{\lambda}) \frac{t_s}{t_h} - \mathcal{H}}{\mathcal{P} + \mathcal{H}} \right\rceil \quad \text{and} \quad \mathcal{N}_{\text{SAIDA}} = \left\lceil n + \frac{(1 - \frac{1}{\lambda}) \frac{t_s}{t_h} - \mathcal{H}}{\mathcal{P} + \mathcal{H}} \right\rceil$$

Nevertheless, Condition (3) must hold. It can be proved that if the above two numbers do not check that equation then none does (the proof relies on the minimality of these integers). So, we define the following two integers and then the gain for each scheme:

$$n_{\text{LY}} = \begin{cases} \left\lceil \frac{\mathcal{P}}{\mathcal{P} + \mathcal{H}} n + \frac{(1 - \frac{1}{\lambda}) \frac{t_s}{t_h} - \mathcal{H}}{\mathcal{P} + \mathcal{H}} \right\rceil & \text{if Condition (3) holds} \\ \text{is not defined} & \text{otherwise} \end{cases}$$

$$n_{\text{SAIDA}} = \begin{cases} \left\lceil n + \frac{(1 - \frac{1}{\lambda}) \frac{t_s}{t_h} - \mathcal{H}}{\mathcal{P} + \mathcal{H}} \right\rceil & \text{if Condition (3) holds} \\ \text{is not defined} & \text{otherwise} \end{cases}$$

The gains are defined as:

$$G_{\text{LY}} = 1 - \frac{n}{n_{\text{LY}}} \quad \text{and} \quad G_{\text{SAIDA}} = 1 - \frac{n}{n_{\text{SAIDA}}}$$

Perrig *et al.* [17] and Pannetrat and Molva [12] attempted to solve two particular cases. They had two different packet sizes: 64 and 512 bytes. We chose the same ones and used MD5 (as hash function) and RSA (as signature scheme). Our results indicate that, when λ is fixed, increasing the number of packet per block n makes the benefit decrease in all cases. This observation is consistent with what we noticed in Section 4.2. Table 3 gives us an approximation of the gain provided by our scheme for $\mathcal{P} = 64$. The value of λ is not specified since it appeared not to have an important impact on the gain.

	10%	25%	50%	75%	90%
G_{LY}	45700	25700	11500	2800	1400
G_{SAIDA}	×	40000	13300	4700	1900

Table 3: Approximation of threshold values about n for G_{LY} and G_{SAIDA} when $\mathcal{P} = 64$.

Our results also showed that when n was small then the gains were close to 1. Remember that λ is fixed (so that Condition (3) holds). When n becomes small (i.e. the block size decreases), Lysyanskaya *et al.*'s technique and SAIDA "tends to" be similar to the sign-each approach scheme (where each packet carries its own signature) whereas our scheme "tends to" be similar a block signature scheme (with λ packets). This justifies the important gain we earn for these values of n . Our observations also indicated that when λ and n were fixed then increasing the packet size \mathcal{P} (from 64 to 512 bytes) made the gain provided by our scheme decrease.

5 Improvements on the Signature Verification Complexity

5.1 Accuracy of the Parameters

The signature verification complexity is $O(\lambda)$ for our scheme which is the same as Lysyanskaya *et al.*'s (when their technique is iterated λ times). We present a modification of our approach which allows to have $O(1)$ instead under some assumptions. We need to introduce the following definition.

Definition 3 We say that a couple (A, B) of survival and flood rates is **accurate** to the network for a flow of N symbols if:

1. Data are sent per block of N elements through the network.

2. For any block of N elements $\{E_1, \dots, E_N\}$ emitted by the sender, if we denote $\{\tilde{E}_1, \dots, \tilde{E}_\mu\}$ the set of received packets then $\mu \leq BN$ and at least AN elements of $\{E_1, \dots, E_N\}$ belong to $\{\tilde{E}_1, \dots, \tilde{E}_\mu\}$.

The second condition must be true for each receiver belonging to the communication group.

Remark: We notice that, when N is fixed, (A, B) is not unique. Indeed, any (\tilde{A}, \tilde{B}) with $\tilde{B} \geq B$ and $0 < \tilde{A} \leq A$ is also accurate for the same flow N .

In our case, we have $N = n$ (see Step 3 of Algorithm 2). We have the following proposition:

Proposition 1 *If (α, β) is accurate then any set of received packet verifies the family signature using $O(1)$ signature verifications.*

Proof.

Denote FID the family number. Assume that we receive a set RP of packets for some block number BID where $BID \in \{1, \dots, \lambda\}$. Since (α, β) is accurate, we have $|\text{RP}| \leq \beta n$ and at least αn packets of RP come from the sender. Denote S_{RP} that subset of RP and C_{RP} the subset of RP consisting of elements having correct numbering (FID, BID, ψ) where $\psi \in \{1, \dots, n\}$. We have: $S_{\text{RP}} \subset C_{\text{RP}} \subset \text{RP}$ so: $\alpha n \leq |S_{\text{RP}}| \leq |C_{\text{RP}}| \leq |\text{RP}| \leq \beta n$.

We will prove that RP verifies the signature by running VerifySignatureFamily with C_{RP} as input. In Step 1, a request to run MGS-Decoder $_\epsilon$ on C_{RP} is executed. The above inequalities prove that C_{RP} is not rejected and a list of size $O(1)$ is output by MGS-Decoder $_\epsilon$. Due to its consistency (see [8]), $h(\tau_1) \parallel \dots \parallel h(\tau_\lambda) \parallel \sigma$ must belong to that list where σ is the signature and $h(\tau_i)$ the hash of the i^{th} original block created by the sender for the family FID. Therefore, Step 2 of VerifySignatureFamily will be successful after at most $O(1)$ signature verifications (Step 3 is not executed). Thus, RP verifies the family signature. □

We deduce the following theorem:

Theorem 2 *If (α, β) is accurate then the complexity of signature verification is $O(1)$.*

Proof.

Denote FID the family number. Assume that we receive a set RP of packets for some block number BID ($BID \in \{1, \dots, \lambda\}$). We run Decoder $_\epsilon$ on RP. Its design indicates that once the family signature has been verified for some set $\widehat{\text{RP}}$, no more signature verifications are performed for FID. Since (α, β) is accurate, we can apply Proposition 1. Therefore, signature verifications for FID are only computed for the first received set $\widehat{\text{RP}}$ for this value FID. Thus, if $\text{RP} = \widehat{\text{RP}}$ then $O(1)$ signature verifications are performed, else no signature verifications are computed. This involves that only $O(1)$ signature verifications are performed for FID. Since no specific values have been assigned to FID, we get our result. □

5.2 Limitations

In practical applications, it is difficult to find a couple (α, β) which is accurate and realistic due to the large number of receivers (potentially several tens of thousands). Using the remark following Definition 3, we can say that if α is "close to" 0 and β "large enough" then (α, β) is accurate for a flow of n . The drawback is that the length of the tag $\tau_{\text{BID}}^i \parallel \tau_\sigma^i$ appended to each packet is approximately $\frac{2}{\rho} \mathcal{H}$ (see Section 3) where $\rho = \frac{\alpha^2}{(1+\epsilon)\beta}$. That is, this tag is around $\frac{\beta}{\alpha^2} (2(1+\epsilon) \mathcal{H})$ large.

If the sender chooses unrealistic values as $\alpha = \frac{1}{10}$ and $\beta = 5$ then we get: $\frac{\beta}{\alpha^2} = 500$. So, the tag is larger than 1000 hashes produced by h (since $\epsilon > 0$). This creates too large an overhead per packet for distribution in the network. The previous values were called "unrealistic" because this choice of (α, β) means that at least 10% of the original packets and a total of no more than $5n$ packets are received. If these values were really accurate then it would mean that the opponent would have a very huge control over the network and the few packets the receivers would authenticate would be probably useless.

If the number of receivers is relatively small then each of them can send back a report of the transmission consisting of his own (α_i, β_i) . Thus, the sender can adjust (α, β) which will be accurate for further transmission as $\alpha = \min_i \alpha_i$ and $\beta = \max_i \beta_i$. Nevertheless, this approach is impracticable when the size of the communication group increases. In this case, the sender has to choose a couple $(\tilde{\alpha}, \tilde{\beta})$ which seems suitable for a large proportion of receivers (for instance 95%) but which is not guaranteed to be accurate for all of them. Therefore, 95% of the receivers will have $O(1)$ as signature verification complexity whereas the other ones will experience $O(\lambda)$ due to potential rejects of received packets by Decoder $_\epsilon$.

6 Conclusion

In [8], Reed-Solomon codes were used to solve the multicast authentication problem in the fully adversarial network. Extending this approach, we designed a scheme where a single signature is computed for every family of λ blocks of n packets. Our technique still allows any receiver to join the communication group at any block boundary. The complexity at the receiver (in term of signature verifications and hash computations) is better than the complexity of λ iterations of Lysyanskaya *et al.*'s protocol. In particular, when the sender has knowledge of an accurate couple (α, β) for most receivers, the complexity of signature verification for a family of λ blocks becomes $O(1)$ (for these participants) which is the complexity of Lysyanskaya *et al.*'s technique for 1 block only.

The minimal value, Λ , of λ such that our extension is the faster one at the sender remains small. For instance, $\Lambda = 2$ up to $n = 30000$ for RSA and MD5. Thus the extra requirements consisting of buffering λn packets is quickly amortized. Since packets are sent and authenticated per block, the throughput of data within the network does not vary too much. Our technique also allows joinability at any block boundary since the family signature is spread into every block. Therefore the size of the communication group can grow even after the beginning of data transmission. The gain provided by our technique is larger than 50% of extra packets per block with respect to SAIDA, eSAIDA, Lysyanskaya *et al.*'s and linear equations protocols up to $n = 11500$. This large value of n should be sufficient for most live applications.

What remains to design is a technique allowing a single signature to be computed for the whole stream respecting the property of joinability. Indeed, in our work, Reed-Solomon codes are used to deal with opponent's malicious actions but we still need to compute one signature every λ blocks to achieve the non-repudiation and joinability property. If such a process exists then it will also be faced with the fully adversarial network model where the opponent can drop a certain amount of chosen data packets.

Acknowledgment

The authors would like to the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by the Australian Research Council under ARC Discovery Project DP0344444. The first author's work was also funded by an iMURS scholarship supported by Macquarie University.

References

- [1] Mohamed Al-Ibrahim and Josef Pieprzyk. Authenticating multicast streams in lossy channels using threshold techniques. In *ICN 2001*, volume 2094 of *Lecture Notes in Computer Science*, pages 239 – 249, Colmar - France, July 2001. Springer - Verlag.
- [2] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology - Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 514 – 532, Gold Coast, Australia, December 2001. Springer - Verlag.
- [3] Yacine Challal, Hatem Bettahar, and Abdelmajid Bouabdallah. A taxonomy of multicast data origin authentication: Issues and solutions. *IEEE Communications Surveys and Tutorials*, 6(3):34 – 57, October 2004.
- [4] Yacine Challal, Abdelmadjid Bouabdallah, and Hatem Bettahar. H₂A: Hybrid hash-chaining scheme for adaptive multicast source authentication of media-streaming. *Computer & Security*, 24(1):57 – 68, February 2005.
- [5] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Advances in Cryptology - Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 180 – 197, Santa Barbara, USA, August 1997. Springer-Verlag.
- [6] Phillippe Golle and Nagendra Modadugu. Authenticating streamed data in the presence of random packet loss. In *Symposium on Network and Distributed Systems Security*, pages 13 – 22, San Diego, USA, February 2001. Internet Society.
- [7] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757 – 1767, September 1999.
- [8] Anna Lysyanskaya, Roberto Tamassia, and Nikos Triandopoulos. Multicast authentication in fully adversarial networks. In *IEEE Symposium on Security and Privacy*, pages 241 – 253, Oakland, USA, May 2003. IEEE Press.
- [9] Florence Jessiem MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- [10] Ralph Merkle. A certified digital signature. In *Advances in Cryptology - Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 218 – 238, Santa Barbara, USA, August 1989. Springer - Verlag.

- [11] Sarah Miner and Jessica Staddon. Graph-based authentication of digital streams. In *IEEE Symposium on Security and Privacy*, pages 232 – 246, Oakland, USA, May 2001. IEEE Press.
- [12] Alain Pannetrat and Rafik Molva. Authenticating real time packet streams and multicasts. In *7th International Symposium on Computers and Communications*, Taormina, Italy, July 2002. IEEE Computer Society.
- [13] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient multicast packet authentication using signature amortization. In *IEEE Symposium on Security and Privacy*, pages 227 – 240, Oakland, USA, May 2002. IEEE Press.
- [14] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient multicast stream authentication using erasure codes. *ACM Transactions on Information and System Security*, 6(2):258 – 285, May 2003.
- [15] Yongsu Park and Yookun Cho. The eSAIDA stream authentication scheme. In *ICCSA*, volume 3046 of *Lecture Notes in Computer Science*, pages 799 – 807, San Diego, USA, April 2004. Springer - Verlag.
- [16] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277 – 292, June 1999.
- [17] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56 – 73, Oakland, USA, May 2000. IEEE Press.
- [18] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2003.
- [19] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [20] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the Association for Computing Machinery*, 36(2):335 – 348, April 1989.
- [21] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *6th ACM Conference on Computer and Communications Security*, pages 93 – 100, Singapore, November 1999. ACM Press.
- [22] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.
- [23] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502 – 513, August 1999.
- [24] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE Conference on Computer Communications*, volume 1, pages 345 – 352, New York, USA, March 1999. IEEE Press.