Lower Bounds for the Signature Size of Incremental Schemes

Marc Fischlin
Fachbereich Mathematik (AG 7.2)/Informatik
Johann Wolfgang Goethe-Universität Frankfurt am Main
Postfach 111932
60054 Frankfurt/Main, Germany
e-mail: marc@mi.informatik.uni-frankfurt.de
URL: http://www.mi.informatik.uni-frankfurt.de/

August 11, 1997

Abstract

We show lower bounds for the signature size of incremental schemes which are secure against substitution attacks and support single block replacement. We prove that for documents of n blocks such schemes produce signatures of $\Omega(n^{1/(2+c)})$ bits for any constant $c > 0$. For schemes accessing only a single block resp. a constant number of blocks for each replacement this bound can be raised to $\Omega(n)$ resp. $\Omega(\sqrt{n})$. Additionally, we show that our technique yields a new lower bound for memory checkers.

1 Introduction

Incremental Cryptography. Suppose that you have a set of documents for which you want to create signatures. Though the documents may only differ in a few positions, with an ordinary signature scheme you usually have to sign each document from scratch. The idea of incremental signature schemes as introduced by Bellare, Goldreich and Goldwasser in [BGG94, BGG95] is that you sign one document M and quickly create the signatures for the other documents from the signature for M and the difference between the documents. More precisely, let $M = M[1] \cdots M[n]$ denote a document of n blocks with signature S. Let $M' = \mathsf{replace}(M, \sigma, i)$ be the document where we replace the i^{th} block $M[i]$ in M by σ. Then an incremental scheme supporting the replace modification allows to produce a signature for M' much faster from M, S, σ and i than it would take to compute a signature from scratch.

Substitution Attacks. The construction of incremental schemes introduces a new kind of attack, so-called substitution attacks. Performing such an attack the adversary is allowed to alter the document before update steps. For example, given a document/signature pair (M, S) the adversary may tamper M to M^* and then call the incremental algorithm to replace a block. As we want the incremental algorithm to be fast it cannot access the whole document M (resp. M^*) but rather read, say, poly$(\log n)$ blocks. Due to this time restriction, the incremental algorithm cannot check the validity of the document/signature pair. Hence, it signs a document different from what it considers to sign.

It has turned out that substitution attacks constitute a very powerful form of attack. Until now, there have only been two construction which are secure against substitution attacks. One construction, IncXMACC, is based on the XOR MACs [BGR95] and can be found in [Fi97]. For documents of n blocks this scheme produces message authentication codes (MACs) of bit length $\Theta(n \log s)$, where s denotes the security parameter. It has the additional property that the incremental algorithm merely accesses the corresponding block for each replacement. We call such schemes single block accessing schemes. The other approach is the tree scheme [BGG95] and its variation [M97]. This scheme produces signatures resp. MACs of length $\Theta(ns)$. The tree scheme remains secure against total substitution attacks, where the adversary may tamper the signature as well as the document before update steps. Beside replacement, both approaches support additional text modifications like single block insertion and deletion.

Our Results. For single block accessing schemes which are secure against substitution attacks and support the replace modification, we show that for all n there exists a document of n blocks such that the incremental scheme produces a signature of size $\Omega(n)$ with positive probability. Thus IncXMACC is optimal up to a factor $O(\log s)$. In fact, a quantitative analysis shows that there are $\Omega(2^{n/4})$ documents of n blocks such that the incremental scheme creates $\Omega(n)$-bit signatures with constant probability for each of these documents. For schemes reading several blocks in every update step, we show a lower bound of $\Omega\left(n^{1/(2+c)}\right)$ for any constant $c > 0$, yet leaving a large gap in comparison to the tree scheme. In this case, the number of documents with n blocks that cause large signatures with probability $\Omega(1/\log n)$ is at least $\Omega(2^{n^{1/(2+c)}/4})$. If the incremental algorithm only reads a constant number of blocks then our technique yields the stronger bound $\Omega(\sqrt{n})$ for the signature size.

For instance, our result shows that we cannot design incremental hash-and-sign schemes that withstand substitution attacks and produce short signatures. An incremental hash-and-sign scheme consists of an incremental hash function and an ordinary signature scheme. The signature for document M is a pair (h_M, S_h), where h_M is the hash value of M and S_h is a signature for h. To update this signature, update the hash value fast by the incremental property and sign the new hash value with the ordinary scheme. We remark that our bounds even hold in the random oracle model [BR93] where signer and adversary share a public random hash function (see discussion in section 3.2). Finally, we prove a new lower bound for noninvasive off-line checkers. See section 5.

Communication Complexity. The lower bounds for the signature size follow by reduction to Yao's randomized two-party communication model [Y79]. Namely, two processors A and B, each one getting a secret input x resp. $y \in \{0,1\}^n$, want to compute $f_n(x,y)$ for a boolean function $f_n : \{0,1\}^{2n} \to \{0,1\}$ by communicating as few bits as possible. It is well known that $\Omega(n)$ bits are necessary for some functions, even if we allow a small error probability. More concretely, througout this paper we'll use the disjointness function disj_n which is 0 if and only if there exists an i such that the i^{th} bits in x and y are both 1. In other words, if we view x, y as the characteristic vectors of two sets, each of n elements, then $\mathrm{disj}_n(x,y) = 1$ iff the sets are disjoint. The lower bound $\Omega(n)$ for disj_n has been shown by Kalyanasundaram and Schnitger [KS92]. Their proof has been simplified by Razborov [R92].

In a nutshell, our construction can be outlined as follows. Processor A signs an appropriate document M_x containing $x \in \{0,1\}^n$ with the incremental scheme and sends this signature to B. Depending on his input y processor B does several incremental signature generations (performing a substitution attack). Then B sends the final signature to A who verifies that this signature is valid for M_x. With high probability this holds if and only if x and y are disjoint. Thus, we compute the disj_n function with small error probability. Let m be the number of blocks in M_x, e.g. we'll use $m = n+1$ and $m = n^{2+c}$. As the communication complexity of disj_n is $\Omega(n)$ we conclude that one of the exchanged signatures must have $\Omega(n)$ bits in the worst case. Substituting the document length m by n yields the lower bound $\Omega(n)$ resp. $\Omega\left(n^{1/(2+c)}\right)$.

Related Work. In [Fi97] we show a lower bound for substitution detecting schemes, where the incremental algorithm verifies in each update step whether relevant parts of the document have been tampered or not. These schemes produce signatures of $\Omega(n/I(n))$ bits, where $I(n)$ denotes the maximum number of accessed blocks for each replacement. This bound has been transfered from the memory checker model of Blum et al. [BEG$^+$94]. For these special schemes, the bound in [Fi97] is tighter than our result, while our bounds are applicable to incremental schemes in general.

Chor, Geréb-Graus and Kushilevitz [CGK95] have already applied lower bounds in communication complexity to obtain impossibility results for so-called private protocols. They show that some functions over the integers cannot be computed privately. Otherwise we could design an efficient protocol for the identity function contradicting lower bounds for the communication complexity of this problem. See [K92, BCKO93] for further applications of the communication model to private protocols. For other aspects of communication complexity we refer the reader to [KN97].

Pedersen and Pfitzmann [PP97] show that the Shannon entropy of fail-stop signatures — and therefore the average bit size in a good approximation — must be, roughly speaking, at least twice the security parameter. Lower bounds for authentication codes have been shown by several authors. See Stinson's book [S95] and Pei's work [P95] for surveys. In contrast to our approach all these lower bounds have been proved by applying combinatorial or entropy techniques directly.

2 Preliminaries

For two strings $x, y \in \{0,1\}^*$ we write $x \cdot y$ or xy for the concatenation of x and y. Denote by x^m the m-fold concatenation of x and let $x_i \in \{0,1\}$ be the i^{th} bit in x.

2.1 Incremental Schemes

We give all definitions and results for digital signature schemes only. We remark that the lower bounds hold for message authentication schemes, too. Documents $M \in \{0,1\}^*$ are divided into blocks $M = M[1] \cdots M[n]$ such that $M[i] \in \Sigma = \{0,1\}^b$. We assume wlog. that every document length is a multiple of b. Otherwise the signature scheme pads each document by some standard technique, e.g., append $10\cdots 0$ to each document for a minumum number of zeros.

For a signature scheme we let s denote the security parameter, k the key size, d the signature key and e the verification key. For simplicity, we assume that b and s are recoverable from k and that $b, k = \text{poly}(s)$. Moreover, we suppose that all documents $M[1]\cdots M[n] \in \Sigma^*$ with $n = \text{poly}(s)$ are admissible.[1] To simplify notation, we give a very succinct definition of incremental signature schemes supporting the replace modification exclusively. See [BGG95, Fi97, M97] for more general definitions.

Definition 2.1 (Incremental Signature Scheme)
A replace*-incremental scheme $\mathcal{S}$ is a tuple* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{IncSig}, \mathsf{Vf})$ *of probabilistic* $\text{poly}(s)$*-time algorithms such that:*

- KGen *generates on input 1^s (the security parameter in unary) a pair of k-bit keys (d, e).*
- *On input an admissible document $M \in \Sigma^*$ and key d, the probabilistic algorithm* Sig *outputs a signature S and stores (M, S).*
- *Given a position i, a block $\sigma \in \Sigma$ and keys d, e, algorithm* IncSig *reads the last stored pair (M, S) and produces a signature S' for $M' = \mathsf{replace}(M, \sigma, i)$ and stores (M', S').*
- *Given a document M, a signature S and key e, the verifier* Vf *returns a bit A with $A = 1$ for "accept" and $A = 0$ for "reject".*

The scheme $\mathcal{S}$ is called complete iff for all keys (d, e) produced with positive probability by KGen *the following two conditions hold: For all admissible documents M and signatures $S = \mathsf{Sig}(d, M)$ we have $\mathsf{Vf}(e, M, S) = 1$ and, secondly, for all admissible documents M and bit strings S such that $\mathsf{Vf}(e, M, S) = 1$ we have $\mathsf{Vf}(e, M', S') = 1$, where $M' = \mathsf{replace}(M, \sigma, i)$ and $S' = \mathsf{IncSig}(d, e, M, S, \sigma, i)$.*

The completeness condition says that signatures which have been produced by Sig from scratch resp. IncSig from a valid document/signature pair must be valid. This requirement can be relaxed by allowing a small error probability. Note that $\mathcal{S}$ can be stateful, i.e., store information about previous signatures and documents. Our lower bounds hold if this information takes only a few bits, say $O(\log s)$, or if the state information can be determined by the number of signature generations (e.g., if $\mathcal{S}$ holds a counter). We'll elaborate this when discussing the protocols. We remark that all known incremental schemes merely use a counter as state information. We omit the keys d, e from the inputs for Sig, IncSig and Vf if they are clear from the context. Furthermore, we abuse notation and say that IncSig gets inputs $\mathsf{replace}(M, \sigma, i)$ and S if the input is σ, i and (M, S) is the stored document/signature pair that IncSig reads.

Our protocols can be easily adapted to work with single block deletion. On the other hand, replace is probably the most simple modification one can think of. This has been confirmed by

[1]It's crucial for our result that the document space contains every polynomial bit string from Σ^* (though $\Sigma = \{0, 1\}$ resp. $b = 1$ suffices). Otherwise we cannot apply the lower bound for the disjointness problem.

the design of all incremental algorithms so far. For instance, if a scheme supports single block insertion and deletion then it also supports replacement. Simply delete the corresponding block and insert the new value.

We want IncSig to be fast. Therefore we assume that this algorithm accesses only a few document blocks for each update step. To be concrete, IncSig can only read $I(n)$ blocks from the stored document $M[1] \cdots M[n]$. Typically, $I(n) = \text{poly}(\log n)$. In this case, we show a lower bound for the signature size of $\Omega\left(n^{1/(2+c)}\right)$ for any constant $c > 0$. Our approach also yields a lower bound of $\Omega\left(n^{d/2}\right)$ for $I(n) \leq n^{1-d}$ for all constants $d > 0$.

An adversary is a family $C = (C_s)_{s \in \mathbb{N}}$ of polynomial size circuits performing an adaptive chosen message attack [GMR88]. That is, for $s \in \mathbb{N}$ the scheme $\mathcal{S}$ is initiated by a pair of keys (d, e) produced by $\mathsf{KGen}(1^s)$. Then adversary C_s queries oracles Sig and IncSig for documents and parameters of his choice, where the i^{th} query depends on e, the coin tosses and the previous $i-1$ signatures and documents. During the attack, the adversary may tamper the last stored document M by an $\mathsf{alter}(M^*)$ command. That is, he replaces M by M^* regardless of the current content of M. However, the signature S and the state information of $\mathcal{S}$ remain unchanged. This form of attack is called a message substitution attack.[2]

We introduce virtual documents [BGG95] to define successful attacks. According to Goldwasser, Micali and Rivest [GMR88], an ordinary signature scheme is broken if the adversary produces a signature for a "fresh" document, i.e., a document that hasn't been signed previously. Considering incremental schemes and substitution attacks, it is not obvious what a "fresh" document is. For instance, if the adversary tampers document M to M^* before calling IncSig then it's not clear which document is supposed to not be viewed as "fresh" anymore. The notion of virtual documents enables us to associate each signature to a document content and, therefore, to designate "fresh" documents. Informally, virtual documents are documents which $\mathcal{S}$ considers to have signed in an attack. More formally, if Sig creates a signature for a document M then the corresponding virtual document $\text{virt}(M)$ is M itself. If IncSig creates a signature for $M' = \mathsf{replace}(M, \sigma, i)$ we let $\text{virt}(M') = \mathsf{replace}(\text{virt}(M), \sigma, i)$. If the adversary issues an $\mathsf{alter}(M^*)$ command for M then we set $\text{virt}(M^*) = \text{virt}(M)$. An adversary is successful if he produces a signature for a document which hasn't appeared as a virtual document before. See [BGG95] for discussions about virtual documents.

A $\mathsf{replace}$-incremental signature scheme is called ϵ-secure against message substitution attacks for $\epsilon : \mathbb{N} \to \mathbb{R}$, if for all adversaries C the probability that C_s performs a successful message substitution attack is at most $\epsilon(s)$ for all $s \in \mathbb{N}$. For the rest of this paper, we assume that $\mathcal{S}$ is a complete, ϵ-secure, $\mathsf{replace}$-incremental scheme with $\epsilon(s) < \frac{c}{2}$ for a constant $c < 1$ and all sufficient large s (so that $\epsilon(s)$ is bounded away from $\frac{1}{2}$ by a constant factor). Note that signature schemes, incremental and ordinary ones, usually achieve the better security level $\epsilon(s) < 1/\text{poly}(s)$ for all sufficient large s (under some standard cryptographic assumption).

[2]This terminology is taken from [Fi97]. To avoid confusion, in this paper we call strings to be signed *documents*, while *messages* are strings being exchanged in the communication model. Thus, we would better say "document substitution attacks".

2.2 Communication Complexity

The communication model consists of two processors A and B, where A gets input $x \in \{0,1\}^n$ and B gets $y \in \{0,1\}^n$. They want to compute the value $f_n(x,y)$ of a boolean function $f_n : \{0,1\}^{2n} \to \{0,1\}$ with low communication cost, i.e., by exchanging as few bits as possible. Both processors share a public random tape and are computationally unbounded. Though in this work, they always run in polynomial time. A protocol P defines the messages that are exchanged. The i^{th} message depends on the first $i-1$ messages, the coin tosses and the corresponding input of the processor. Let A give the output bit. Denote by $P(x,y) \in \{0,1\}$ the random variable describing the output distribution of the protocol for inputs x, y. Here, the probability is taken over the public coin tosses.

Define $\text{err}(f_n, P) = \max\{\text{Prob}[P(x,y) \neq f_n(x,y)] \mid x,y \in \{0,1\}^n\}$ to be the error probability of P. Let $\text{C}(P,x,y)$ be the maximum number of bits communicated by protocol P for inputs x, y, where the maximum is taken over all random strings. Then $\text{C}(P) = \max\{\text{C}(P,x,y) \mid x,y \in \{0,1\}^n\}$. For $0 \leq \delta < \frac{1}{2}$ define the bounded error communication complexity for f_n by

$$R_\delta^{\text{pub}}(f_n) = \min\{\text{C}(P) \mid P \text{ is protocol with } \text{err}(f_n, P) \leq \delta\}$$

We have $R_\delta^{\text{pub}}(\text{disj}_n) = \Omega(n)$ for all constants δ with $0 \leq \delta < \frac{1}{2}$ by [KS92, R92]. The superscript "pub" indicates that the communication complexity is defined in the public coin model (as opposed to the private coin model [KN97]). In Appendix A we present an introduction to other notions of the communication complexity of a Boolean function, which we need in sections 3.3 and 3.4.

3 A Lower Bound for Single Block Accessing Schemes

In this section we show a lower bound for replace-incremental signature schemes where IncSig merely reads the j^{th} block for a $\text{replace}(\cdot,\cdot,j)$ command. We assume that $n = n(s) = \text{poly}(s)$ is a fixed function of the security parameter s. Then we can identify the function disj_n for parameter n and the adversary C_s for security parameter s.

The key observation for the protocol is that $\text{disj}_n(x,y) = 1$ if and only if $x_j = 0$ for all $j \in \{1,2,\ldots,n\}$ with $y_j = 1$. Party A signs x and sends the signature to B.[3] For all positions j with $y_j = 1$ party B calls IncSig to replace the j^{th} bit by 0 producing a sequence of at most n signatures. In each update step, B guesses that the original bit x_j was 0. That is, B first substitutes x by 0^n. Since IncSig only reads the corresponding block for each update step, substituting x by 0^n in fact means to exactly alter those bits x_j with $y_j = 1$ to 0. Then B generates the incremental signatures as described above and finally sends the last signature to A, who outputs 1 if and only if this signature is valid for x.

If $\text{disj}_n(x,y) = 0$, or equivalent if $x_j = 1$ for some j with $y_j = 1$, then B has altered the corresponding bit. As $\mathcal{S}$ is secure against these attacks the signature won't be valid for x with high probability, i.e., $P(x,y) = 0$. On the other hand, if $\text{disj}_n(x,y) = 1$ resp. $x_j = 0$

[3]To be precise, A signs $x \cdot 0$. We omit the technical nuisances in this informal, introductory description.

for all j with $y_j = 1$, then B's guesses have been correct and since $\mathcal{S}$ is complete, we have $P(x, y) = 1$.

3.1 The Protocol

Wlog. suppose that the block size b is one. Otherwise pad each bit with $b - 1$ zeros. We augment x and y by a bit '0' to obtain documents $X = x \cdot 0$ and $Y = y \cdot 0$, each of $n+1$ blocks resp. bits. We use this extra bit to show that intersecting inputs yield a successful attack on the incremental scheme for a document which hasn't appeared as a virtual document.

A and B seperately run $\mathsf{KGen}(1^s)$ with their common random tape to generate the same pair (d, e) of k-bit keys. Then A signs X with Sig to obtain a signature S_A. A sends this signature to B. Now B works as follows. Let $J = \{j \in \{1, \dots, n\} \mid y_j = 1\}$. For notational convenience, we assume that J is arbitrarily ordered and that J_i denotes the i^{th} element in J. Furthermore, we let $Z = 0^{n+1}$. Then

$$\mathrm{disj}_n(x, y) = 1 \quad \text{iff} \quad X_j = Z_j = 0 \text{ for all } j \in J. \tag{1}$$

B alters X to Z. Note that B doesn't need to know X for this. Let $S^{(0)} = S_A$.

For $h = 0, \dots, |J|-1$ algorithm B runs IncSig for $\mathsf{replace}(Z, 0, J_{h+1})$ on signature $S^{(h)}$ to obtain a new signature $S^{(h+1)}$. Observe that $Z = \mathsf{replace}(Z, 0, J_{h+1})$, i.e., Z remains unchanged by this operation. Finally, B sets the last bit in Z by calling IncSig for $\mathsf{replace}(Z, 1, n+1)$ and $S^{(|J|)}$ to obtain a signature S_B. Then B sends this signature S_B to A. Receiving signature S_B processor A runs Vf for $x \cdot 1$ and S_B and outputs 1 iff Vf accepts.

A and B share the same signature scheme $\mathcal{S}$. This construction works, for example, if the state information of $\mathcal{S}$ is predictable by the number of signature generation, i.e., the state information consists of a counter. Then we can assume wlog. that B always makes $n + 1$ update steps (e.g., by repeating $\mathsf{replace}(Z, 0, J_1)$ modifications). Therefore, A and B know the state of $\mathcal{S}$ when receiving a signature. Alternatively, we can communicate the state information with each signature. In this case, our lower bounds remain valid up to a constant factor as long as the bit length of the state information is $\mathrm{o}(n)$. For example, $\mathcal{S}$ could recycle random bits. In this case the coin tosses are part of the state information. All algorithms of $\mathcal{S}$ are polynomial time. Hence, the number of random bits that have been used in A's or B's phase can be written down with $O(\log n)$ bits. Communicating this number with each message suffices because A and B share a common random string. We remark that the state information usually is part of the signature because we cannot expect the verifier to know the previous signature generations. Hence, if it is too large then we will have long signatures anyway.

In our protocol, we call IncSig to replace a bit '0' by '0' again. It is reasonable to assume that IncSig refuses to produce a new signature in this case. Fortunately, we can easily fix this by first replacing the bit '0' by '1' and then resetting the bit to '0' again. Hence, for the analysis we presume that the incremental algorithm supports this kind of replacement.

3.2 Analysis

We show that the protocol computes disj_n with small error probabilty.

Lemma 3.1 (Completeness)
If $\mathrm{disj}_n(x,y) = 1$ *then* $P(x,y) = 1$.

Proof. Since $\mathrm{disj}_n(x,y) = 1$ we have $X_j = Z_j = 0$ for all $j \in J$ by equation (1). Thus, running IncSig for $\mathsf{replace}(Z, 0, J_{h+1})$ is equivalent to running IncSig for document $X = \mathsf{replace}(X, 0, J_{h+1})$ for all h. Here we exploit the fact that IncSig merely reads the corresponding block for update steps. Because $X_{n+1} = Z_{n+1} = 0$, this holds also for the last step where B replaces the augmented bit '0' by '1'. As $\mathcal{S}$ is complete we obtain $\mathsf{Vf}(x \cdot 1, S_B) = P(x,y) = 1$. ■

Lemma 3.2 (Soundness)
If $\mathrm{disj}_n(x,y) = 0$ *then* $P(x,y) = 1$ *with probability at most* $\epsilon(s)$.

Proof. Let $C = (C_s)_{s \in \mathbb{N}}$ be a family of polynomial size circuits. Then C_s gets as non-uniform polynomial advice x, y with $\mathrm{disj}_n(x,y) = 0$ such that x, y maximize the probability that $P(x,y) = 1$. Circuit C_s simulates the protocol using its oracle access to Sig and IncSig (after these algorithms have been initiated with a random pair of keys). Obviously, this can be done in polynomial size.

We show that C_s produces a valid forgery with the single Vf query given that $P(x,y) = 1$. From equation (1) we obtain that $X_j = 1$ for some $j \in J$. The virtual documents appearing in the execution are $W^{(0)} = X$, $W^{(h+1)} = \mathsf{replace}(W^{(h)}, 0, J_{h+1})$ for $h = 0, \dots, |J| - 1$ and $W = \mathsf{replace}(W^{(|J|)}, 1, n+1)$. The latter one is the only virtual document having '1' at position $n+1$. But $W_i = 0$ for all $i \in J$ and therefore $W_j = 0 \neq 1 = X_j = x_j$. Thus, $x \cdot 1$ hasn't appeared as a virtual document. By assumption, $P(x,y) = \mathsf{Vf}(x \cdot 1, S_B) = 1$. Hence, this is a successful message substitution attack on $\mathcal{S}$ which happens with probability at most $\epsilon(s)$. ■

Observe that our construction works in the random oracle model [BR93], too. In this case, we have a public random hash function which we can simulate using the public random tape.

Theorem 3.3
For every n *there exists a document* M *of* $n+1$ *blocks such that* Sig *or* IncSig *produces with positive probability a valid signature for* M *with bit size* $\Omega(n)$.

Note that substituting the document length $n+1$ by n doesn't change the asymptotic bound.

Proof. Let δ be a constant with $\epsilon(s) < \delta < \frac{1}{2}$ for large s. We know that $R_\delta^{\mathrm{pub}}(\mathrm{disj}_n) = \Omega(n)$. Assume towards contradiction that the signature length of all documents with $n+1$ blocks is less than $\frac{1}{2} R_\delta^{\mathrm{pub}}(\mathrm{disj}_n)$ (minus the size for the state information). Then protocol P communicates less than $R_\delta^{\mathrm{pub}}(\mathrm{disj}_n)$ bits. By Lemma 3.1 and 3.2, the protocol computes disj_n with error probability at most $\epsilon(s) < \delta$ and we derive a contradiction.

So far we've only proved that (the valid) signature S_A for X or signature S_B has $\Omega(n)$ bits. However, S_B might not be valid. But we can conclude that there must be x, y such that S_A has at least $\Omega(n)$ bits or $\mathrm{disj}_n(x,y) = 1$ and therefore S_B is a valid signature with $\Omega(n)$

bits. If this were not the case, then for all constants c the following would hold for infinitely many n: For all $x, y \in \{0,1\}^n$ with $\mathrm{disj}_n(x, y) = 1$ we would communicate less than cn bits. Thus, we could design a protocol that stops after having exchanged at most cn bits, because then A and B would already know that $\mathrm{disj}_n(x, y) = 0$. This would be a contradiction to the lower bound for disj_n. ■

Theorem 3.3 only states that the incremental scheme produces signatures with $\Omega(n)$ bits *with positive probability for one document.* However, we remark that for all known schemes the signature size does not depend on the specific document nor the random bits (up to a constant factor, see [M97] for an example where the size varies). Anyhow, the quantitative analysis in section 3.4 shows that there are $\Omega\left(2^{n/4}\right)$ documents for which the incremental scheme produces large signatures with constant probability.

3.3 From Non-Uniform to Uniform Adversaries

We have defined security of signature schemes in terms of non-uniform adversaries. This enables us to "wire" an appropriate pair $(x, y) \in \{0,1\}^{2n}$ into circuit C_s. Razborov [R92] has proved the lower bound for $R_\delta^{\mathrm{pub}}(\mathrm{disj}_n)$ by presenting a distribution ρ on $\{0,1\}^{2n}$ for which the *distributional complexity* (see [KN97] or Appendix A) is $\Omega(n)$. For simplicity let $n = 4L - 1$. Then x, y are generated as follows:

- Choose a random partition $(T_x, \{i\}, T_y)$ of $\{1, \ldots, n\}$ such that $|T_x| = |T_y| = 2L - 1$.
- Let x and y be random sets of size L drawn from $T_x \cup \{i\}$ and $T_y \cup \{i\}$, respectively.

Clearly, we can sample x, y in polynomial time (in n). Observe that each of the inputs x and y has $L \leq \lceil n/4 \rceil$ bits '1'. If $n(s)$ is efficiently computable by a Turing machine on input 1^s, then we can replace circuit family $(C_s)_{s\in\mathbb{N}}$ by an uniform adversary. Note that the error probability of our protocol is still $\epsilon(s)$. However, it is not straightforward that the communication complexity remains $\Omega(n)$, because the probability space now consists of the distribution of the random tape *and* the input. But the lower bound for the distributional complexity holds in this case as well (see [KN97]). Hence, even incremental signature schemes which are "only" secure against uniform adversaries produce large signatures.

3.4 A Quantitative Version of Theorem 3.3

In this section we analyze the number of documents that cause large signature and the probability for it. The quantitative theorem 3.4 below remains valid in the uniform setting.

Theorem 3.4
For every n there are $\Omega\left(2^{n/4}\right)$ documents of n blocks such that for each of these documents, Sig *or* IncSig *produces a valid signature with bit size $\Omega(n)$ with constant non-zero probability.*

In particular, the average signature size for these documents is $\Omega(n)$. The proof of theorem 3.4 is given in Appendix C. We investigate the ratio of documents with large signatures. Party

A signs a document X with L blocks '1'. Hence there are $\binom{n}{L}$ possibilities. Party B signs the real document $0^n \cdot 1$ and virtual documents having L or $L-1$ bits '1'. Thus, the total number of (real and virtual) documents for which we produce signatures S_A, S_B is at most [GKP89,ch.9,ex.42]

$$1 + 2 \cdot \binom{n}{L} + \binom{n}{L-1} \leq 1 + 9 \cdot \binom{n}{\lfloor n/4 \rfloor} \leq 2^{nH(1/4) - \frac{1}{2}\log_2 n + O(1)}.$$

where $H(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$ is the binary entropy function. Since $H(1/4) \leq .812$ there are at most $O\left(2^{.812n}\right)$ possible documents while we have $\Omega\left(2^{.25n}\right)$ documents for which $\mathcal{S}$ produces large signatures with constant probability.

4 A General Lower Bound

In this section we show that every incremental scheme accessing at most $I(m)$ blocks in update steps for documents with m blocks must produce signatures of $\Omega\left(\ell^{-1}(n)\right)$ bits for all invertible functions $\ell(n)$ satisfying

$$\frac{n^2 \cdot I(\ell(n))}{\ell(n)} < c_\ell \tag{2}$$

for a positive constant $c_\ell < \ln 2$ and all sufficient large n. For example, if $I(m) = O(\log^d m)$ for constant d then $\ell(n) = n^{2+c}$ resp. $\ell^{-1}(n) = n^{1/(2+c)}$ satisfies equation (2) for any constant $c > 0$:

$$\frac{n^2 \cdot I(\ell(n))}{\ell(n)} = \frac{O((2+c)^d \cdot \log^d n)}{n^c} \xrightarrow{n \to \infty} 0.$$

Note that for $c = 0$ this bound would be $\ell^{-1}(n) = \sqrt{n}$. A more careful analysis shows that $\ell(n) = cn^2 \log^d n$ for an appropriate constant c is sufficient. In this case we have $\ell^{-1}(n) = \Omega\left((n/\log^d n)^{1/2}\right)$. If $I(m) = O(m^{1-d})$ for a constant $d > 0$ we may set $\ell^{-1}(n) = \Omega(n^{d/2})$. For $I(m) \leq d$ let $\ell^{-1}(n) = \sqrt{\frac{n}{2d}} = \Omega\left(\sqrt{n}\right)$.

To motivate our construction we explain why we cannot apply the idea from the previous section directly. There, B assumed that all x_j with $y_j = 1$ must be 0. Otherwise we have $x_j = y_j = 1$ and an error occurs (unless we break the incremental scheme). Here, IncSig might read additional blocks to update the signature. The major problem is that B would have to guess all the values IncSig reads. More precisely, assume that B wants to replace block j with $y_j = 1$ by 0. Let y_h be another block that IncSig would read for this update step and suppose that $y_h = 0$. Then B has to guess x_h. If he's wrong (e.g., if $x_h = 0$ but B tries 1) then the final signature S_B won't be valid with high probability though the inputs might yet be disjoint. Note that bits $y_h = 1$ don't cause problems. For these bits, B tries $x_h = 0$ which will be correct if x, y are disjoint and leads to the desired message substitution otherwise. This example shows that the "dangerous" bits are those with $y_i = 0$. For these bits, B will guess by default that the corresponding bit x_i is 1. Hence we'll only be wrong for blocks with $x_i = y_i = 0$.

We sketch the underlying idea of our solution. We stretch x, y by an expansion factor $E(n) = \frac{\ell(n)}{n}$ by appending $\ell(n) - n$ blocks '0'. Denote these extended documents by X and Y. Using their common coin tosses A and B agree on a random permutation π over $\{1, \dots, \ell(n)\}$. Let $\pi(X)$ and $\pi(Y)$ denote the documents obtained by applying π to X resp. Y. See figure 1. Note that $\mathrm{disj}_n(x, y) = \mathrm{disj}_{\ell(n)}(\pi(X), \pi(Y))$. Then we adapt the protocol of the previous section for $\pi(X)$ and $\pi(Y)$. Namely, A signs $\pi(X)$ and sends the signature S_A to B. Party B iteratively calls the incremental algorithm to replace all blocks in $\pi(X)$ such that a bit $y_i = 1$ has been mapped to the corresponding position in $\pi(Y)$. By the choice of $\ell(n)$ and as the bits with $x_i = y_i = 0$ are distributed in a substring of $\pi(X)$ of size at least $\ell(n) - n$ the incremental algorithm IncSig won't pick such a block with high probability.

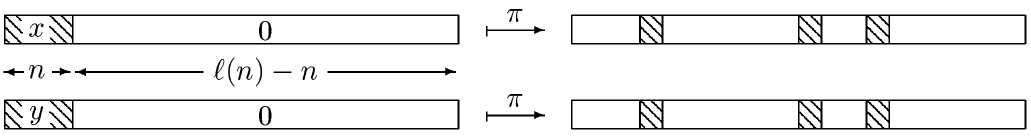

Figure 1: Expanding and Permuting the Documents

4.1 The Protocol

Let $X = x \cdot 0^{\ell(n)-n}$ and $Y = y \cdot 0^{\ell(n)-n}$. A and B use their public coin flips to jointly generate a random permutation π over $\{1, 2, \dots, \ell(n)\}$. We write $\pi(X)$ for $X_{\pi^{-1}(1)} X_{\pi^{-1}(2)} \cdots X_{\pi^{-1}(\ell(n))}$.

A signs $\pi(X)$ with Sig and sends the signature S_A to B. Again, let $J = \{j \mid y_j = 1\}$ and J_i be the i^{th} element in J (for some fixed order). Denote $\{\pi(J_i) \mid i = 1, \dots, |J|\}$ by $\pi(J)$. B sets $Z \in \{0,1\}^{\ell(n)}$ according to

$$Z_i = \begin{cases} 0 & \text{if } \pi^{-1}(i) \in \{n+1, \dots, \ell(n)\} \cup J \\ 1 & \text{else} \end{cases}$$

Then Z equals $\pi(X)$ on bits $\pi(n+1), \dots, \pi(\ell(n))$ and is set to $1 - y_i$ for $\pi^{-1}(i) \in \{1, \dots, n\}$. Party B alters $\pi(X)$ to Z. Let $S^{(0)} = S_A$. For $h = 0, \dots, |J| - 1$ algorithm B runs IncSig on $\mathsf{replace}(Z, 0, \pi(J_{h+1}))$ and signature $S^{(h)}$ to obtain $S^{(h+1)}$. We have again $Z = \mathsf{replace}(Z, 0, \pi(J_{h+1}))$ for all h. Finally, let S_B be the signature produced by IncSig for $\mathsf{replace}(Z, 1, \pi(n+1))$ and $S^{(|J|)}$. Receiving signature S_B, party A runs Vf for X' and S_B where $X' = \mathsf{replace}(\pi(X), 1, \pi(n+1))$ and outputs 1 iff Vf accepts.

4.2 Analysis

Assume that A and B have agreed on a permutation π. Let $H = \{h \in \{1, \dots, n\} \mid x_h = 1\}$ and $J = \{j \in \{1, \dots, n\} \mid y_j = 1\}$. Note that $H \cap J = \emptyset$ if and only if x, y are disjoint. We say that a block i in $\pi(X)$ is *forbidden* with respect to (x, y, π) iff $\pi^{-1}(i) \in \{1, 2, \dots, n\} \setminus (H \cup J)$. Informally, a block is forbidden if some x_h with $x_h = y_h = 0$ has been mapped to that position. A protocol execution is called *good* iff IncSig doesn't access a forbidden block with respect to

(x, y, π). If the execution is good then all blocks $i \notin \pi(J)$ in $\pi(X)$ which IncSig reads equal Z_i. In this case, whether running IncSig for Z or $\pi(X)$ is equivalent or not, only depends on the values x_j for $j \in J$. We can therefore apply the idea of the previous section.

We start by showing that the execution is good with high probability. We abbreviate $E(n)$ by E and $I(\ell(n))$ by I. The proof of the following Lemma is given in Appendix B.

Lemma 4.1
The probability that an execution is good is at least $\exp\left(-\frac{nI}{E-I-1}\right)$ *for all sufficient large* n.

By equation (2) we have $\exp\left(-\frac{nI}{E-I-1}\right) > \frac{c}{2}$ for a constant $c > 1$ and all sufficient large n, i.e., the probability is bounded away from $\frac{1}{2}$ by a constant factor.

Lemma 4.2 (Completeness)
If $\mathrm{disj}_n(x, y) = 1$ *then* $P(x, y) = 1$ *with probability at least* $\exp\left(-\frac{nI}{E-I-1}\right)$ *for all sufficient large* n.

Proof. Suppose that the execution is good. This happens with the claimed probability. First observe that party B only replaces blocks $\pi(j)$ in $\pi(Y)$ with $y_j = 1$ and we have $x_j = 0$ for all such j by assumption $\mathrm{disj}_n(x, y) = 1$. Additionally, other blocks that IncSig reads equal the corresponding values in Z because the execution is good. Now the claim follows as in Lemma 3.1. ■

Lemma 4.3 (Soundness)
Let $\ell(n) = \mathrm{poly}(n)$. *If* $\mathrm{disj}_n(x, y) = 0$ *then* $P(x, y) = 1$ *with probability at most* $\epsilon(s)$.

Proof. The proof is similar to Lemma 3.2. Since $\ell(n) = \mathrm{poly}(n)$ circuit C_s can simulate the protocol in polynomial size. Note that we don't rely on a good execution. ■

Now we substitute the document length $\ell(n)$ by $\ell^{-1}(\ell(n)) = n$. The proof of the following theorem is similar to the proof of theorem 3.3.

Theorem 4.4
Let $\ell(n) = \mathrm{poly}(n)$ *be an invertible function satisfying equation* (2). *Then for every* n *there exists a document* M *of* n *blocks such that* Sig *or* IncSig *produces with positive probability a valid signature for* M *with bit size* $\Omega\left(\ell^{-1}(n)\right)$.

The quantitative version can be derived as before. Though, we were only able to show a lower bound of $\Omega(1/\log n)$ for the probability that Sig or IncSig produces a large signature.

Theorem 4.5
Let $\ell(n) = \mathrm{poly}(n)$ *be an invertible function satisfying equation* (2). *Then for every* n *there are* $\Omega(2^{\ell^{-1}(n)/4})$ *documents of* n *blocks such that for each of these documents,* Sig *or* IncSig *produces a valid signature with bit size* $\Omega\left(\ell^{-1}(n)\right)$ *with probability* $\Omega(1/\log n)$.

In this case, the number of documents with large signatures is $\Omega(2^{.25\ell^{-1}(n)})$ compared to the total number of $O\left(2^{.812n}\right)$ documents. With the same technique as in section 3.3 we can replace circuit family C by an uniform adversary if $n(s)$ and $\ell(n)$ are polynomial-time computable. The proof of theorem 4.5 appears in Appendix D. It requires a rigorous analysis because we now have a distribution on the documents, too. This distribution is due to the random permutation we apply to the stretched inputs. In Appendix E we sketch how to improve the bounds for schemes where IncSig has a very simple access strategy.

5 Lower Bounds for Memory Checkers

Consider the data structure RAM (random access memory) with readable and writeable bit cells. If we store values via RAM on an insecure memory they could be tampered by an adversary causing read operations to give wrong answers. A memory checker provides a method to detect such errors. Instead of giving our operations to RAM directly we pass it to the checker who cares about storage and retrieval. To verify the correctness of the output of RAM the checker can keep information in his private memory which the adversary cannot read or tamper. A checker is called invasive if it stores additional information on the insecure memory. Otherwise it is called noninvasive. An on-line checker gives a warning immediately when an error occurs, while off-line checkers may output "error" at a later point of the execution. More formal definitions are given in [BEG$^+$94] and [Fi97].

Blum et al. [BEG$^+$94] give lower bounds for the size of the checker's private memory for RAM$_n$, i.e., random access memory with n cells. Namely, they show that (invasive and therefore noninvasive) off-line checkers must have $\Omega(\log n)$ bits private memory. For noninvasive on-line checkers they prove the lower bound $\Omega(n/I(n))$ where $I(n)$ denotes the maximum number of cells which the checker reads for each operation. This bound can be easily transfered to substitution detecting incremental signature schemes [Fi97]. To best of our knowledge there haven't been proved any lower bounds for noninvasive off-line checker so far (except for the bound $\Omega(\log n)$ for invasive checkers).

We briefly describe how to apply our technique to the memory checker model. We'll deal with the general case of arbitrary access strategies. Our improvements for special access methods of incremental signature schemes hold in the checker scenario as well. Additionally, our quantitative theorems can be easily adapted. Assume that we are given a noninvasive off-line checker for RAM. Processor A runs the memory checker writing $\pi(X)$ into memory. Let $M_A \in \{0,1\}^{m_A}$ denote the checker's private memory after this sequence of operations. Then A sends M_A and the number r_A of random bits used by the checker with $m_A + O(\log r_A)$ bits to B. Note that the checker's state is totally determined by M and the coin tosses. Party B continues the checker's simulation by reading all values $\pi(j)$ with $y_j = 1$. B uses the same string Z as in section 4 to predict zhe values that the checker reads. Finally, B sends the content $M_B \in \{0,1\}^{m_B}$ of the checker's private memory and the number r_B of random bits to A (and a bit indicating whether the checker has already given an error message or not). Processor A passes "finished" to the checker that may give a few more operations to RAM in a postprocessing phase. A simulates these additional operations using M_B and $\pi(X)$. It outputs that x, y are disjoint if and only if the checker hasn't returned a warning. The same analysis as in section 4 shows that the error probability of this protocol is sufficiently small. Hence the memory size of the checker can be $\Omega\left(\ell^{-1}(n) - \log r(n)\right)$ where $r(n)$ denotes the number of random bits used by the checker. Clearly, this bound holds for on-line checkers, too, but it is inferior to the one given in [BEG+94].

6 Open Problems

There is still a gap between the signature size of known incremental schemes and our lower bounds. We've only considered message substitutions. An interesting question is whether we can exploit the notion of total substitution attacks to prove stronger bounds or not. Another interesting point are time-space-tradeoffs. We've proved stronger bounds for schemes accessing only a constant number of blocks in each update step. Can one derive similar results for example for $I(m) = \log m$? On the other hand, we've proved stronger bounds for simple access strategies of IncSig. Can one show improved bounds for these access methods?

Acknowledgements

We thank Georg Schnitger for very helpful discussions. We also thank Daniele Miccianco and the FOCS committee for valuable comments.

References

[BCKO93] R.Bar-Yehuda, B.Chor, E.Kushilevitz, A.Orlitsky: Privacy, Additional Information, and Communication, *IEEE Information Theory, Vol. 39(6), pp.* 1930–1943, 1993.

[BGG94] M.Bellare, O.Goldreich, S.Goldwasser: Incremental Cryptography: The Case of Hashing and Signing, *Crypto '94, Lecture Notes in Computer Science, Vol. 839, Springer-Verlag, pp.* 216–233, 1994.

[BGG95] M.Bellare, O.Goldreich, S.Goldwasser: Incremental Cryptography and Application to Virus Protection, *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing, pp.* 45–56, 1995.

[BGR95] M.Bellare, R.Guérin, P.Rogaway: XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions, *Crypto '95, Lecture Notes in Computer Science, Vol. 963, Springer-Verlag, pp.* 15–29, 1995.

[BR93] M.Bellare, P.Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *ACM Conference on Computer and Communication Security, pp.* 63–72, 1993.

[BEG+94] M.Blum, W.Evans, P.Gemmell, S.Kannan, M.Naor: Checking the Correctness of Memories, *Algorithmica, Volume 12, pp.* 225–244, 1994.

[CGK95] B.Chor, M.Geréb-Graus, E.Kushilevitz: Private Computations Over the Integers, *SIAM Journal on Computing, Vol. 24(2), pp.* 376–386, 1995.

[Fe68] W.Feller: An Introduction to Probability Theory and its Applications (Volume 1), *Addison-Wesley Publishing Company*, 1968.

[Fi97] M.Fischlin: Incremental Cryptography and Memory Checkers, *Eurocrypt '97, Lecture Notes in Computer Science, Vol. 1233, Springer-Verlag, pp.* 393–408, 1997.

[GMR88] S.Goldwasser, S.Micali, R.L.Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks, *SIAM Journal on Computation, Vol. 17(2), pp.* 281–308, 1988.

[GKP89] R.L.Graham, D.E.Knuth, O.Patashnik: Concrete Mathematics, *Addison-Wesley Publishing Company*, 1989.

[KS92] B.Kalyanasundaram, G.Schnitger: The Probabilistic Communication Complexity of Set Intersection, *SIAM Journal on Discrete Mathematics, Vol. 5(4), pp.* 545–557, 1992.

[K92] E.Kushilevitz: Privacy and Communication Complexity, *SIAM Journal on Discrete Mathematics, Vol. 5(2), pp.* 273–284, 1992.

[KN97] E.Kushilevitz, N.Nisan: Communication Complexity, *Cambridge University Press*, 1997.

[M97] D.Micciancio: Oblivious Data Structures: Application to Cryptography, *Proceedings of the 29th Annual Symposium on the Theory of Computing*, 1997.

[PP97] T.Pedersen, B.Pfitzmann: Fail-Stop Signatures, *SIAM Journal on Computing, Vol. 26(2), pp.* 291–330, 1997.

[P95] D.Pei: Information-Theoretic Bounds for Authentication Codes and Block Designs, *Journal of Cryptology, Vol. 8, pp.* 177–188, 1995.

[R92] A.A.Razborov: On the Distributional Complexity of Disjointness, *Theoretical Computer Science, Vol. 106, pp.* 385–390, 1992.

[S95] D.Stinson: Cryptography — Theory and Practice, *CRC Press*, 1995.

[Y79] A.C.Yao: Some Complexity Questions Related to Distributive Computing, *Proceedings of the 11th Annual Symposium on the Theory of Computing, pp.* 209–213, 1979.

A On the Notion of Communication Complexity

We give a very brief introduction to the relevant aspects of communication complexity. See [KN97] for a comprehensive survey.

We first define two other measures of the communication complexity of a Boolean function. In section 2.2 we have defined the bounded error complexity of a protocol with respect to the maximum over the coin tosses. In contrast to this worst case definition, the *average complexity* of a protocol averages the communication over the random coin tosses. Namely, let $\mathrm{C}(P, w, x, y)$ be the number of bits which P exchanges for inputs x, y and fixed coin tosses w. Then

$$\mathrm{C}^{\mathrm{AC}}(P, x, y) = \sum_{w} \mathrm{Prob}[\omega = w] \cdot \mathrm{C}(P, w, x, y)$$

where ω describes the distribution of the coin flips. The average communication complexity of P is $\mathrm{C}^{\mathrm{AC}}(P) = \max\left\{ \mathrm{C}^{\mathrm{AC}}(P, x, y) \;\middle|\; x, y \in \{0,1\}^n \right\}$. Again, let $\mathrm{err}(f_n, P)$ denote the error probability of P. Then the average bounded error communication complexity of a Boolean function f_n is

$$R_\delta^{\mathrm{AC,pub}}(f_n) = \min\left\{ \mathrm{C}^{\mathrm{AC}}(P) \;\middle|\; P \text{ is protocol with } \mathrm{err}(f_n, P) \leq \delta \right\}$$

In both cases, the error probability has been defined with respect to the internal coin tosses of the parties. Now we consider deterministic protocols with random inputs. Let μ_n be a distribution on inputs $\{0,1\}^{2n}$ and let D be a deterministic protocol. We say that D has error probability δ with respect to f_n and μ_n iff $\mathrm{Prob}[D(x, y) \neq f_n(x, y)] \leq \delta$, where the probability is taken over the distribution of (x, y) according to μ_n. The communication complexity of D is the maximum over all inputs x, y (which occur with positive probability). The *distributional complexity* $\mathrm{Distr}_\delta^{\mu_n}(f_n)$ of a function f_n and distribution μ_n is the minumum over all deterministic protocols D with error probability δ with respect to f_n and μ_n.

For notational convenience we will abbreviate the different types of protocols by *average case*, *worst case* and *distributional* protocols and write P^{AC}, P^{WC} and D. If we talk about protocols in general we write P. Furthermore, we abbreviate distribution μ_n on $\{0,1\}^{2n}$ by μ if n is clear from the context.

If we view a worst case protocol P^{WC} as an average case protocol P^{AC}, then the error probability is preserved and the communication complexity of P^{AC} for inputs x, y is bounded above by that of P^{WC} for x, y. On the other hand, if an average case protocol P^{AC} communicates $g(x, y)$ bits for inputs x, y on the average, then we can derive a protocol P^{WC} that exchanges $O(g(x, y))$ bits in the worst case and has the same error probability for this input. Thus, $R_\delta^{\mathrm{pub}}(f_n) = \Theta\big(R_\delta^{\mathrm{AC,pub}}(f_n)\big)$.

For every distribution μ, every randomized protocol P^{WC} with error probability δ (over the coin tosses) can be easily turned into a deterministic one D with error probability δ (over the random inputs) by fixing an appropriate sequence ω of coin flips. Moreover, the communication complexity of D is bounded above by the complexity of P^{WC}. Hence,

$$R_\delta^{\mathrm{pub}}(f_n) \geq \max_{\mu} \mathrm{Distr}_\delta^{\mu}(f_n)$$

and it's not hard to show equality [KN97].

B Proof of Lemma 4.1

Consider the following randomized game between IncSig and our permutation generating algorithm. In this game, IncSig tries to find a forbidden block while the generator picks a random permutation to hide these blocks in $\pi(X)$. Observe that after the i^{th} update step IncSig hasn't obtain more information about the random permutation than it gets from the state information about $\pi(X)$ sent by A (including the signature for $\pi(X)$), the blocks that it has yet read from $\pi(X)$ and the query positions $\pi(J_1),\dots,\pi(J_i)$. In support of the signature scheme we will assume that IncSig is given $\pi(X)$ and $\pi(J)$ in advance. Note that we even handicap the generator by this.

IncSig reads at most nI bits from $\pi(X)$. We say that IncSig is successful if it reads a forbidden block, i.e., finds a position $h \in \{1,\dots,\ell(n)\}$ with $\pi^{-1}(h) \in \{1,\dots,n\} \setminus (H \cup J)$. Obviously, IncSig maximizes its success probability by choosing different blocks in $L = \{i \in \{1,\dots,\ell(n)\} \mid X_{\pi^{-1}(i)} = 0\} \setminus \pi(J)$. We have $|L| \geq \ell(n) - |H| - |J|$ (with equality iff x, y are disjoint). The forbidden blocks are uniformly distributed among L. Moreover, IncSig's view (including $\pi(X)$ and $\pi(J)$) gives no advantage in predicting their posititions. We conclude that the probability that IncSig doesn't access a forbidden block is distributed hypergeometric [Fe68]. Namely, we have a population of at least $\ell(n)-|H|-|J| = nE-|H|-|J|$ elements of which at most n are red (i.e. forbidden bits) and at least $nE-n$ are black (padded bits or blocks with $x_i = 1$ or $y_i = 1$). Then IncSig randomly draws a set of (at most) nI elements from the population.[4] It accesses a forbidden block if and only if it picks a red element.

Let the random variable R denote the number of red elements when R is distributed according to the description above. Then the probability that IncSig doesn't access a forbidden block in the whole execution is given by the probability for $R = 0$:

$$\begin{aligned}
\text{Prob}[\,R=0\,] &\geq \frac{\binom{nE-n}{nI}}{\binom{nE-|H|-|J|}{nI}} \geq \frac{(nE-n)\cdot(nE-n-1)\cdots(nE-n-nI+1)}{nE\cdot(nE-1)\cdots(nE-nI+1)} \\
&= \left(1-\frac{n}{nE}\right)\cdot\left(1-\frac{n}{nE-1}\right)\cdots\left(1-\frac{n}{nE-nI+1}\right) \\
&\geq \left(1-\frac{1}{E-I}\right)^{nI}
\end{aligned}$$

From $1-\frac{1}{x} \geq \exp\left(-\frac{1}{x-1}\right)$ for $x > 1$ we derive

$$\text{Prob}[\,R=0\,] \geq \exp\left(-\frac{nI}{E-I-1}\right).$$

■

[4] Actually, IncSig does only $\lceil n/4 \rceil + 1$ replacements, because we can restrict the inputs $x, y \in \{0,1\}^n$ to have at most $\lceil n/4 \rceil$ bits '1' according to Razborov's distribution (section 3.3).

C Proof of Theorem 3.4

Recall from Appendix A that we denote average case, worst case and distributional protocols by P^{AC}, P^{WC} and D, respectively. Protocols in general are denoted by P. For protocol P we call inputs $x, y \in \{0,1\}^n$ *bad* if x, y are disjoint and P communicates $\Omega(n)$ bits (where the communication complexity is defined in the corresponding setting). For disjoints inputs x, y the signatures S_A, S_B refer to documents $x \cdot 0$ and $x \cdot 1$. Hence, the number of *"bad" documents* merely depends on A's input x. This motivates the following definition. Let

$$\mathcal{B}_n^P = \{x \in \{0,1\}^n \mid (x,y) \text{ is bad for } P \text{ for some } y \in \{0,1\}^n\}$$

denote the set of A's inputs that participate in bad input pairs. Then we are interested in the size of $\mathcal{B}_n^P$. In accordance with the notation above, we call $x \in \{0,1\}^n$ *bad* for P iff $x \in \mathcal{B}_n^P$. In particular, when we say that a protocol P gets a bad input $x \in \mathcal{B}_n^P$ it is understood that P gets an input (x, y) such that (x, y) is bad for P.

The proof of theorem 3.4 is divided into three parts. First, by analyzing the distributional complexity of disj_n for Razborov's distribution $\rho = \rho_n$ (see section 3.3) we show that the size of $\mathcal{B}_n^D$ for each distributional protocol D is at least $\Omega\left(2^{n/4}\right)$. Then, using that every average case protocol can be transformed into a worst case one and that this protocol can then be turned into a distributional one, we conclude that the size of $\mathcal{B}_n^{P^{\mathrm{AC}}}$ for each average case protocol P^{AC} is $\Omega\left(2^{n/4}\right)$. Finally, we prove that this does not only hold with respect to average complexity, but that $\Omega(n)$ bits must be communicated with constant non-zero probability for inputs $x \in \mathcal{B}_n^{P^{\mathrm{AC}}}$.

Lemma C.1
Let D be a deterministic protocol with error probability $\delta < \frac{1}{2}$ and let $\overline{\delta} = \frac{1}{4} - \frac{1}{2}\delta$. Then with probability at least $\overline{\delta}$ (over the random choice of the inputs) the input $(x, y) \in \{0,1\}^{2n}$ is bad for D.

Proof. Assume that for all constants c there is a protocol D^c with error probability δ such that this protocol communicates more than cn bits for disjoint inputs with probability less than $\overline{\delta}$ for infinitely many n. From D^c we construct a deterministic protocol $\overline{D}^c$ which achieves error probability $\delta + \overline{\delta} < \frac{1}{2}$ and exchanges at most cn bits for infinitely many n. $\overline{D}^c$ simulates D^c untill the output has been given or cn bits have been communicated. In the latter case, $\overline{D}^c$ outputs that x, y intersect. The error probability and communication complexity of $\overline{D}^c$ can be easily verified. But this contradicts the lower bound for the distributional complexity of disj_n. □

Lemma C.2
Let $n = 4L-1$ and let x, y, $T = (T_x, i, T_y)$ denote the random variables distributed according to Razborov's distribution ρ. Let x_0, y_0 be disjoint elements from $\{0,1\}^n$ and $T_0 = (T_x^0, \{i_0\}, T_y^0)$ be a fixed partition of $\{1, \ldots, n\}$ with $|T_x^0| = |T_y^0| = 2L - 1$. Then

$$\mathrm{Prob}[\,(x,y) = (x_0, y_0) \mid T = T_0\,] \leq 9 \cdot \binom{2L}{L}^{-2}$$

Proof. Given the partition $(T_x^0, \{i_0\}, T_y^0)$ the distribution of x, y only depends on the random choice from sets $T_x^0 \cup \{i_0\}$ and $T_y^0 \cup \{i_0\}$. With probability $\frac{3}{4}$ the variables x, y will be disjoint. In this case, there are three possibilities concerning i_0. Either i_0 isn't element of both x and y, or i_0 belongs to exactly one set. In the former case, x and y are random sets of size L, each one drawn independently from a population of size $2L-1$. In the latter case, each of the sets of size L resp. $L-1$ is drawn independently from a population of size $2L-1$. Thus,

$$\mathrm{Prob}[\,(x,y) = (x_0,y_0)\ |\, T = T_0\,] \le \frac{3}{4} \cdot \left[\binom{2L-1}{L}^{-2} \;+\; 2 \cdot \binom{2L-1}{L}^{-1} \binom{2L-1}{L-1}^{-1} \right]$$

The claim follows from $\binom{2L-1}{L-1} = \binom{2L-1}{L} = \frac{1}{2} \cdot \binom{2L}{L}$. □

Combining the two lemmata we obtain:

Corollary C.3
For each deterministic protocol D with error probability δ (over the choice of the inputs) there are $\Omega\left(2^{n/4}\right)$ bad $x \in \mathcal{B}_n^D$.

Proof. Let $\overline{\delta} = \frac{1}{4} - \frac{1}{2}\delta$. By lemma C.1, input (x, y) is bad with probability at least $\overline{\delta}$. To simplify notation, we abbreviate $(T_x, \{i\}, T_y)$ and $(T_x^0, \{i_0\}, T_y^0)$ by T and T_0, respectively. Furthermore, let $T_{y,i_0}^0(L)$ denote the set of subsets of $T_y^0 \cup \{i_0\}$ having L elements. Similar for $T_{x,i_0}^0(L)$.

$$\begin{aligned}
\overline{\delta} &\le \mathrm{Prob}[\,(x,y) \text{ is bad}] \\
&\le \sum_{T_0} \sum_{y_0 \in T_{y,i_0}^0(L)} \sum_{\substack{x_0 \in \mathcal{B}_n^D \cap T_{x,i_0}^0(L) \\ \mathrm{disj}_n(x_0,y_0)=1}} \mathrm{Prob}[\,(x,y) = (x_0,y_0)\ |\, T = T_0\,] \cdot \mathrm{Prob}[T = T_0] \\
&\le \sum_{T_0} \binom{2L}{L} \sum_{x_0 \in \mathcal{B}_n^D} 9 \cdot \binom{2L}{L}^{-2} \cdot \mathrm{Prob}[T = T_0] \\
&\le \sum_{T_0} \sum_{x_0 \in \mathcal{B}_n^D} 9 \cdot \binom{2L}{L}^{-1} \cdot \mathrm{Prob}[T = T_0] \\
&\le \sum_{x_0 \in \mathcal{B}_n^D} 9 \cdot \left(\frac{2L}{L}\right)^{-L} \\
&\le 9 \cdot 2^{-n/4} \cdot \left|\mathcal{B}_n^D\right|
\end{aligned}$$

□

Next we show that the number of bad x is preserved under average complexity:

Lemma C.4
For each randomized protocol P^{AC} with error probability δ there are $\Omega\left(2^{n/4}\right)$ bad $x \in \mathcal{B}_n^{P^{\mathrm{AC}}}$.

Proof. Transform the average case protocol P^{AC} into a worst case one P^{WC} such that the error probability is preserved. Note that P^{WC} communicates $\Omega(n)$ bits in the worst case for the same inputs x, y as P^{AC} does on the average. Then turn P^{WC} into a distributional protocol D for ρ by fixing a suitable sequence ω of coin flips. By Corollary C.3 there are $\Omega\left(2^{n/4}\right)$ bad $x \in \mathcal{B}_n^D$. But then there must be $\Omega\left(2^{n/4}\right)$ bad x for P^{WC} and hence for P^{AC}. □

We prove that $\Omega(n)$ bits are necessary not only on the average, but with constant non-zero probability for each $x \in \mathcal{B}_n^P$. Informally, this comes from the fact that $\Omega(n)$ bits are necessary and $O(n)$ bits are sufficient.

Lemma C.5
For each randomized protocol P with error probability δ there are $\Omega\left(2^{n/4}\right)$ bad $x \in \mathcal{B}_n^P$ such that for each of these bad inputs x protocol P communicates $\Omega(n)$ bits with constant non-zero probability.

Proof. Let P be an average case protocol. Note that worst case protocols can be transformed into average case protocols. Wlog. we can assume that P communicates at most $O(n)$ bits for each input.[5] More concretely, let dn be an upper bound for the communication of P for constant $d > 1$.

Fix an arbitrary bad input (x, y) and let cn be a lower bound for $\mathrm{C}^{\mathrm{AC}}(P, x, y)$. Suppose that the probability that P exchanges more than $d^{-1}cn$ bits for x, y is less than $\frac{1-1/d}{d}c$. Then

$$\begin{aligned}
cn &\leq \mathrm{C}^{\mathrm{AC}}(P,x,y) = \sum_{w} \mathrm{Prob}[\omega = w] \cdot \mathrm{C}(P,w,x,y) \\
&= \sum_{\substack{w \\ \mathrm{C}(P,w,x,y) > d^{-1}cn}} \mathrm{Prob}[\omega = w] \cdot \mathrm{C}(P,w,x,y) \\
&\qquad + \sum_{\substack{w \\ \mathrm{C}(P,w,x,y) \leq d^{-1}cn}} \mathrm{Prob}[\omega = w] \cdot \mathrm{C}(P,w,x,y) \\
&< \tfrac{1-1/d}{d} c \cdot dn + d^{-1}cn \\
&= cn
\end{aligned}$$

and we derive a contradiction. □

This completes the proof of theorem 3.4. ■

D Proof of Theorem 4.5

The proof is similar to the proof in Appendix C. In comparison to the proof there, we now have a distribution on the documents as well (implied by the random permutation). While

[5] Otherwise modify P so that if more than n bits have been communicated, then party A sends his whole input to B.

the documents have n blocks, the underlying disjointness problem has input size $\ell^{-1}(n)$. To distinguish between these cases, we write P_n for a protocol that gets inputs $x, y \in \{0,1\}^n$ and $P_{\ell^{-1}(n)}$ for inputs $x, y \in \{0,1\}^{\ell^{-1}(n)}$. Let ρ denote Razborov's distribution. We write $P^{\rho}_{\ell^{-1}(n)}$ for a randomized protocol that gets random inputs distributed according to ρ. Let $P^{\pi\circ\rho}_n$ denote a randomized protocol that gets n bits inputs, where the inputs are distributed according to ρ and then stretched to n bits and permuted by a random permutation.

Assume that there is a protocol P_n with error probability δ that communicates $\Omega\left(\ell^{-1}(n)\right)$ bits (in the worst case) for less than $\Omega(2^{\ell^{-1}(n)/4})$ bad inputs $x \in \{0,1\}^n$. Then P_n achieves the same error probability and communication complexity for each input (x, y) if we distribute the inputs according to $\pi \circ \rho$. But then we can easily derive a randomized protocol $P^{\rho}_{\ell^{-1}(n)}$ with input distribution ρ which contradicts the number of bad inputs from corollary C.3. That is, for input x, y the two parties in protocol $P^{\rho}_{\ell^{-1}(n)}$ apply the same stretch-and-permute technique as in section 4. Then, they run $P^{\pi\circ\rho}_n$ and give the same output. Obviously, protocol $P^{\rho}_{\ell^{-1}(n)}$ computes $\mathrm{disj}_{\ell^{-1}(n)}$ with P_n's error probability. But then there must be $\Omega(2^{\ell^{-1}(n)/4})$ bad x for P_n. In addition, we conclude that this bound holds in the average complexity setting, too.

Next, we prove a lower bound for the probability that the incremental scheme produces a long signature for each of these bad inputs x. The proof is similar to the proof of lemma C.5. Clearly, we have the upper bound $O\left(\ell^{-1}(n) \cdot \log n\right)$ for the communication complexity, because if more than $\ell^{-1}(n) \cdot \log n$ bits have been communicated then A simply sends all the bit positions of '1' to B. The claim follows as in lemma C.5 by substituting cn by $c \cdot \ell^{-1}(n)$, dn by $d \cdot \ell^{-1}(n) \cdot \log n$ and $\frac{1-1/d}{d}c$ by $\frac{1-1/d}{d \log n}c$. ■

E Improved Lower Bounds for Special Access Strategies

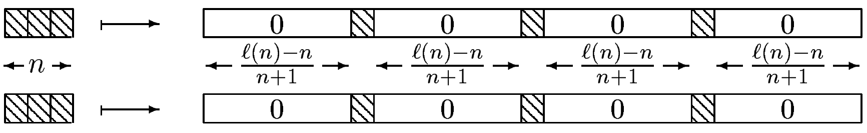

Figure 2: Expanding the Documents if IncSig reads only adjacent blocks

Recall that we've used the stretch-and-permute technique because IncSig may choose $I(m)$ blocks randomly. We briefly discuss how to derive stronger bounds if IncSig's access has a special form. Assume for example that for replace(M, σ, j) commands with $M = M[1] \cdots M[m]$, IncSig reads only blocks which are at most $N(m)$ positions to the left or right from j. Let $\ell(n)$ be an invertible function with

$$N(\ell(n)) < \frac{\ell(n) - n}{n+1}$$

for large n. We expand documents x and y to $\ell(n)$ bits by inserting $(\ell(n) - n)/(n+1)$ bits '0' in front of every x_i resp. y_i and behind the n^{th} bit. See figure 2. This time we don't have to permute the extended documents. Obviously, we can now apply our method and derive lower bounds $\ell^{-1}(n) = \Omega(n)$ for $N(m) \leq d$ and $\ell^{-1}(n) = \Omega(n/\log^d n)$ for $N(m) = O(\log^d m)$. One can easily show analogous results for similar access strategies.