Application-Specific Cryptographic Schemes Based on Symmetric-Key Primitives

Shoichi Hirose

University of Fukui, Japan

ASK 2014
(2014/12/19-22, SETS, Chennai)

Outline

1. Background and Motivation

2. Redactable Signature Scheme for Tree-Structured Data

3. Forward-Secure Sequential Aggregate Message Authentication

Joint work with Hidenori Kuwakado

Background and Motivation

Almost all cryptographic protocols/schemes use hash functions.

$$H : \{0,1\}^* \to \{0,1\}^n$$

Security requirements for hash functions:

- (Second-)Preimage resistance (Onewayness)
  H is easy, and H^{-1} is difficult.
- Collision resistance
  It is difficult to find distinct M, M' s.t. $H(M) = H(M')$.
- Random oracle
  H is a random function.
- Pseudorandom function (PRF, for keyed hash functions)
  H_K is indistinguishable from a random function.

Background and Motivation

Problems

- Random oracle is an ideal assumption.
- There exists a large gap between OW and CR [Simon 98]:
  - A CR HF cannot be constructed with a black-box OW permutation.

Important to identify requirements for hash functions

- Needs multiple requirements?
- Really needs RO?
- Really needs CR?

Redactable Signature Scheme for Tree-Structured Data Based on Merkle Tree

- Background
- Related Work
- Definition
- Proposed Scheme
- Provable Security

Background

Database outsourcing with clouds

- Owners of data outsource database service to a provider.

Security requirements

- Confidentiality of data
  Unauthorized users should not have access
- Correctness proofs of answers to queries

Problem

- Efficient processing of encrypted data is difficult
- Unreasonable to prepare signatures of all possible answers in advance
  - Queries are various
  - Access rights are different from users

Useful if signature of data by owner is redactable by provider

Related Work

Early work on redactable signature

[Steinfeld, Bull, Zheng 2001] Content extraction signature

[Miyazaki et al. 2003] Digital document sanitization

- The owner
  1. divides documents into parts
  2. signs commitments of all parts of a document
- The provider reveals some parts to users
  - according to their access rights
  - without using owner's signing key

Related Work

Redactable signature for tree-structured data

For tree-structured data and its signature,
signatures of subtrees are computable without the signing key

[Kundu, Bertino 2008] First scheme, turned out insecure

[Bruzska et al. 2010]

- Formal definitions of security requirements
- Scheme using ordinary signature

[Samelin et al. 2012], [Pöhls et al. 2012]

- Allow more flexible redaction
  Eg.: Removal of internal node(s)

These schemes are inefficient:

- Signing requires $\Omega(N)$ calls to ordinary signing.
- N: Number of nodes of the tree

Our Contribution

Redactable signature scheme for tree-structured data

- Based on Merkle tree
- Signing involves only one call to ordinary signing procedure
- Provably secure

☹ The proposed scheme can be applied to tree-structured data with

Out-degree $\leq$ constant (chosen by application)

Redactable Signature Scheme for Tree-Structured Data

$\mathsf{tSig} = (\mathsf{tK}, \mathsf{tS}, \mathsf{tV}, \mathsf{tC})$

Key generation $(sk, pk) \leftarrow \mathsf{tK}(1^\ell)$

ℓ is a security parameter

Signing $(T, \sigma) \leftarrow \mathsf{tS}(sk, T)$

σ is a signature for tree-structured data T

Verification $d \leftarrow \mathsf{tV}(pk, T, \sigma)$

$$d = \begin{cases} 1 & \text{if } \sigma \text{ is a valid signature for } T \text{ w.r.t. } pk \\ 0 & \text{otherwise} \end{cases}$$

Cutting $(T', \sigma') \leftarrow \mathsf{tC}(pk, T, \sigma, L)$

L is a leaf of T

σ' is a signature for $T' = T \backslash L$

The secret signing key sk is not used

Multiple cutting produces signature of any sub-tree sharing the root with T

Security Requirements of Redactable Signature for Tree

[Bruzska et al. 2010]

Unforgeability

Similar to EUF-CMA of ordinary signature

Existential UnForgeability against adaptive Chosen Message Attacks

Difference: Redaction is not forgery

Transparency

Formalized by impossibility to tell whether a signature is created

- only by signing, or
- by first signing, and then cutting

Impossible to tell whether cutting is carried out or not after signing

- No information is leaked on the deleted parts (if any)

Unforgeability

$\mathcal{A}$	Adversary	tK	Key generation algorithm
		tS	Signing algorithm

Experiment

$(sk, pk) \leftarrow \mathsf{tK}(1^\ell)$

$(T, \sigma) \leftarrow \mathcal{A}^{\mathsf{tS}(sk,\cdot)}(pk)$ $\quad\triangleright$ *Let* $T_1, T_2, \ldots, T_q$ *be queries to* tS *by* $\mathcal{A}$

if $($*σ is a valid signature for* $T) \wedge ($*T is not a sub-tree of* $T_i)$ **then**

Success in forgery

else

Failure in forgery

Unforgeable $\Leftrightarrow \Pr[$Success in forgery$] =$ negligible

Transparency

$\mathcal{A}$	Adversary	tK	Key generation algorithm
		tS	Signing algorithm
		tC	Cutting algorithm

Experiment

$(sk, pk) \leftarrow \mathsf{tK}(1^\ell)$
$b \leftarrow \{0, 1\}$
$d \leftarrow \mathcal{A}^{\mathsf{tS}(sk,\cdot),\mathsf{SorC}(\cdot,\cdot,sk,b)}(pk)$
if $d = b$ **then**
    Success
else
    Failure

function SorC(T, L, sk, b)
    if $b = 0$ **then**
        $(T, \sigma) \leftarrow \mathsf{tS}(sk, T)$
        $(T', \sigma') \leftarrow \mathsf{tC}(pk, T, \sigma, L)$
    else
        $T' \leftarrow T \backslash L$
        $(T', \sigma') \leftarrow \mathsf{tS}(sk, T')$
    return (T', σ')

Transparent $\Leftrightarrow \left|\Pr[\text{Success}] - 1/2\right|$ is negligible

Proposed Scheme (Signing Algorithm)

H hash function
K master secret key (for transparency)
r nonce

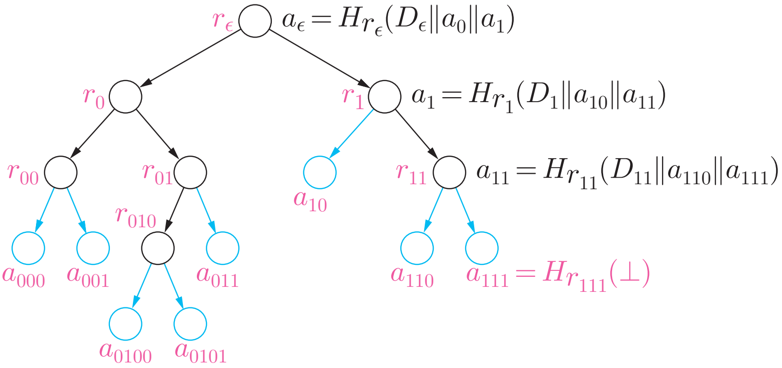
Let out-degree $\leq d$

1. (Construction of Merkle tree) For a given tree T,
   1. Construct tree T' by adding dummy child nodes and edges for nodes (including leaves) with out-degree $< d$
   2. For each node v_i of T', compute secret key $r_i = H_K(r \| i)$
   3. Construct Merkle tree using H_{r_i} for node v_i
2. Sign the root digest using an ordinary signature scheme

Signing Algorithm (with Example of Merkle Tree, out-deg. ≤ 2)

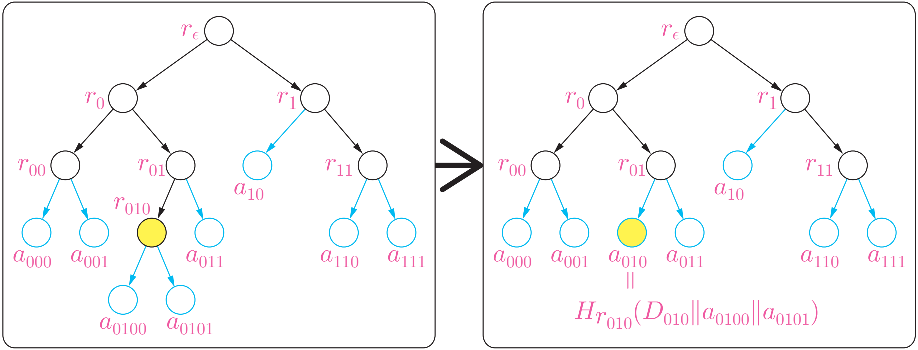
The signature of T (drawn with black) is a tuple of

- Signature of the root digest a_ϵ
- Digests $a_i = H_{r_i}(\perp)$ of all dummy nodes (drawn with blue)
- Secret keys $r_i = H_K(r\|i)$ corresponding to nodes v_i of T



Cutting Algorithm (Example)

The leaf v_{010} (yellow) is cut: v_{010} becomes a dummy node.



New signature is obtained by

1. removing secret key r_{010} and digests a_{0100}, a_{0101} from the original
2. adding $a_{010} = H_{r_{010}}(D_{010}\|a_{0100}\|a_{0101})$

Provable Security of Proposed Scheme

tSig	proposed scheme
Sig	ordinary signature scheme for root digest
H	hash function

Theorem (Unforgeability)

(Sig is unforgeable) $\wedge$ *(H is collision resistant)* $\Rightarrow$ tSig *is unforgeable*

- Unforgeability of Sig avoids forgery of signature for new root digests.
- CR of H avoids generation of Merkle trees with the same root digest.

Theorem (Transparency)

Keyed mode of H is a pseudorandom function $\Rightarrow$ tSig *is transparent*

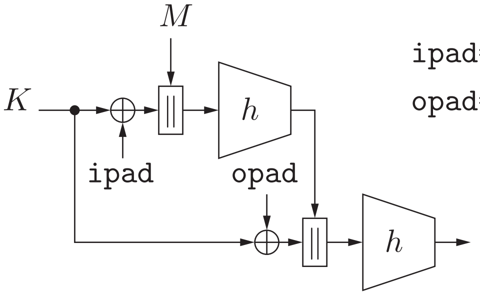
- Digests of nodes look random due to the PRF property of H_K.

HMAC

HMAC can instantiate the HF H in the proposed scheme:

- Used as a pseudorandom function
- Hash function h is collision-resistant (CR) $\Rightarrow$ HMAC is CR

MAC (Message Authentication Code) function using a hash function



ipad=0x363636...

opad=0x5c5c5c...

Conclusion

Redactable signature scheme for tree-structured data

- Based on Merkle tree using keyed hash function such as HMAC
    - efficient, but
    - out-degree $\leq$ const
- Provable security (unforgeability & transparency)
- Extension to DAG (Directed Acyclic Graph) is straightforward.

Future work

Efficient & provably secure scheme for

- more general tree
- graph

Forward-Secure Sequential Aggregate Message Authentication Revisited

- Background
- Related Work
- Definition
- Proposed Scheme
- Provable Security

Background

Message authentication

- MAC function F should be unforgeable

$$
\begin{array}{ccc}
\text{Sender} & & \text{Receiver} \\
 & \xrightarrow{(M_1, \tau_1)} & \\
\tau_i = F_K(M_i) & & \tau_i = F_K(M_i)\ ? \\
 & \xrightarrow{(M_2, \tau_2)} & \\
 & \vdots &
\end{array}
$$

Applications such as secure logging and sensor networks require

- forward secrecy (for the case of secret-key leakage)
- detection of reordering and deletion
- reduction of resource consumption (memory, transmission power, ...)

Forward-Secure Sequential Aggregate Message Authentication

FS SAMA [Ma, Tsudik 2007]

Forward Secure

- Impossible to forge tags for keys before leakage
- Achieved by secret-key update

Sequential

- Reordering and deletion are detectable

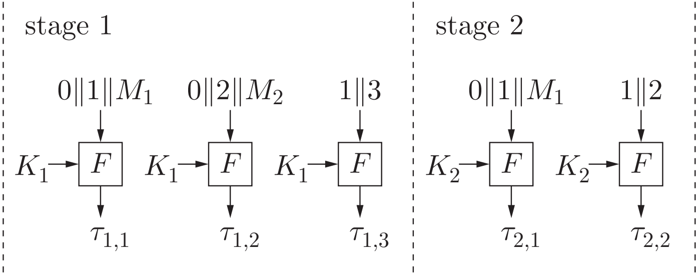
Aggregate

- Tags for messages are aggregatable
- Single tag for a sequence of messages

Related work

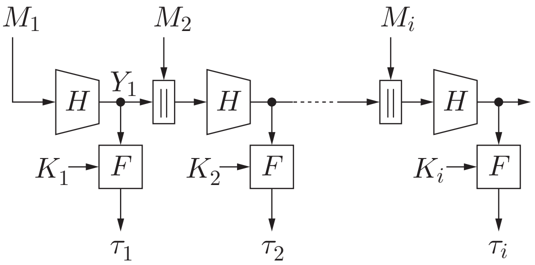
- Forward secure message authentication for audit logs
  [Bellare, Yee 1997], [Schneier, Kelsey 1999]
- History-free message authentication
  [Eikemeier et al. 2010]

Bellare, Yee 1997



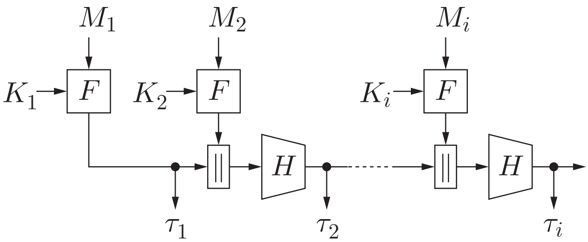
- Numbering scheme
- F is a MAC function.
- K_i is used during stage i.
- Reordering and deletion are detected by
  - message-numbering,
  - end-marker.
- Aggregation is not considered.

Schneier, Kelsey 1999



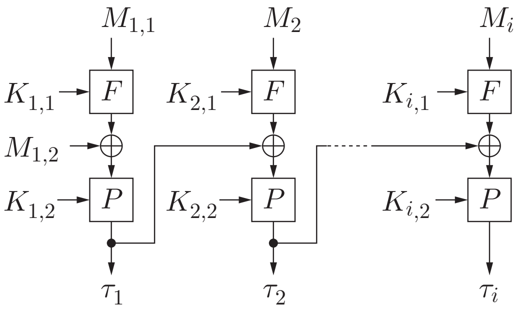
- Linking scheme
- F is a MAC function.
- H is a collision-resistant hash function.
  It is difficult to find distinct X, X' such that $H(X) = H(X')$.
- The secret key is updated after each tagging operation.
- Aggregation is possible.
  τ_i is a tag for $(M_1, \ldots, M_i)$.

Ma, Tsudik 2007



- Linking scheme
- F is a MAC function.
- H is a collision-reisitant hash function.
- The secret key is updated after each tagging operation.
- Aggregation is possible.

Eikemeier et al. 2010



- Linking scheme
- F is a MAC function.
- P is a PRP (pseudorandom permutation)
- The keys for F are independent of the keys for P.
- More flexible aggregation is possible.
- Forward secrecy is not considered.

Our Contribution

- Formalization of scheme and security
- New scheme without CR HF and PRP
- Reduction of the security of the scheme to
  - indistinguishability of the key generator, and
  - unforgeability or indistinguishability of the MAC function

Comparison with previous schemes

Scheme	Aggregation	Col. Resis.	PRP	ProvSec
Bellare-Yee	☹	✓	✓	✓
Schneier-Kelsey	✓	☹	✓	?
Ma-Tsudik	✓	☹	✓	?
Eikemeier et al.	✓✓	✓	☹	✓
Ours	✓	✓	✓	✓

FS SAMA: Definition (1/2)

SAM $=$ (kgen, update, tag, verif, aggre, n), n is the number of stages

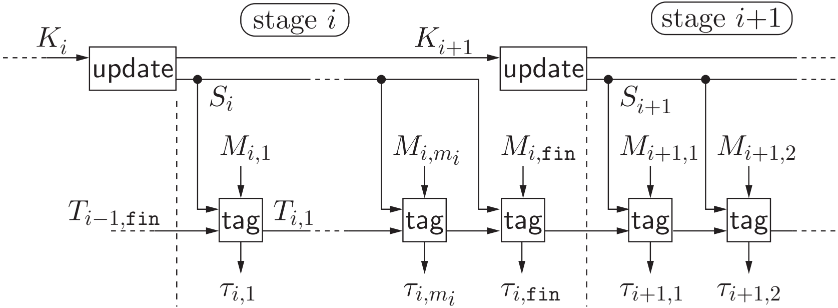
Key Generation $K_1 \leftarrow$ kgen(1^ℓ), ℓ is a security parameter.

Key Update $(S_i, K_{i+1}) \leftarrow$ update(K_i) ($1 \leq i \leq n$)

- S_i is a key for tagging during the i-th stage.

Tagging $(\langle \tau_{i,j}, i \rangle, T_{i,j}) \leftarrow$ tag($S_i, T_{i,j-1}, M_{i,j}$) ($1 \leq i \leq n$)

- $\tau_{i,j}$ is a tag for message $M_{i,j}$.
- $T_{i,j}$ is a state.



FS SAMA: Definition (2/2)

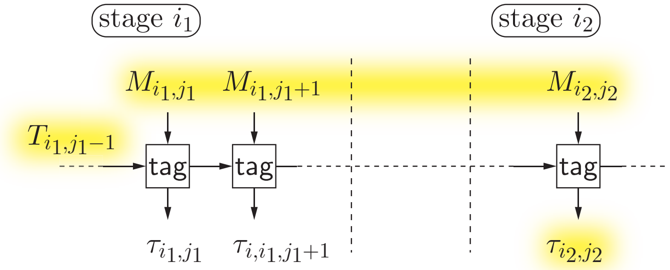
Verification $\alpha \leftarrow \mathsf{verif}(S_{[i_1,i_2]}, T_{i_1,j_1-1}, \boldsymbol{M}_{[(i_1,j_1),(i_2,j_2)]}, \langle \tau_{i_2,j_2}, i_2 \rangle)$

- $\boldsymbol{M}_{[(i_1,j_1),(i_2,j_2)]} = (M_{i_1,j_1}, \ldots, M_{i_2,j_2})$ is a sequence of messages.

Aggregation $(T_{i_1,j_1-1}, \boldsymbol{M}_{[(i_1,j_1),(i_2,j_2)]}, \langle \tau_{i_2,j_2}, i_2 \rangle)$

$$\leftarrow \mathsf{aggre}(T_{i_1,j_1-1}, \boldsymbol{M}_{[(i_1,j_1),(i_2,j_2)]}, \boldsymbol{\tau}_{[(i_1,j_1),(i_2,j_2)]})$$

- Considers aggregation across stages
- Straightforward from the verification algorithm
- $\boldsymbol{\tau}_{[(i_1,j_1),(i_2,j_2)]} = (\langle \tau_{i_1,j_1}, i_1 \rangle, \ldots, \langle \tau_{i_2,j_2}, i_2 \rangle)$ is a sequence of tags for $\boldsymbol{M}_{[(i_1,j_1),(i_2,j_2)]}$.

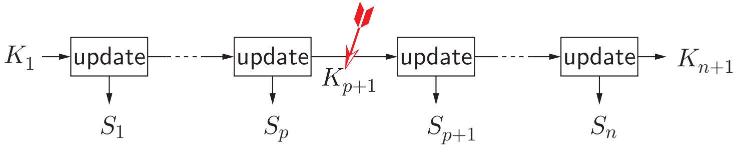


FS SAMA: Definition of Security

$\mathrm{Exp}^{\text{fs-samac}}_{\mathsf{SAM},\mathcal{A}}$

Adversary $\mathcal{A}$

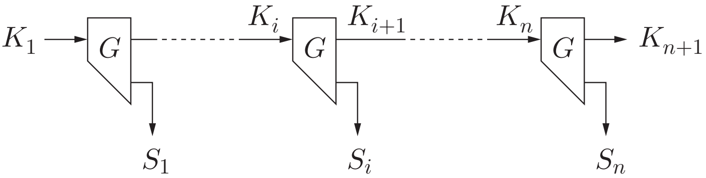
1. (Up to the p-th stage) ▷ $\mathcal{A}$ is allowed to choose p arbitrarily.
   1. $(S_i, K_{i+1}) \leftarrow \mathsf{update}(K_i)$
   2. Makes queries to $\mathsf{tag}(S_i, \cdot, \cdot)$ and gets pairs of a message and a tag.
2. Obtains K_{p+1}.
3. Produces a pair of message sequence and tag for key S_i with $i \leq p$.



$$\mathrm{Adv}^{\text{fs-samac}}_{\mathsf{SAM}}(\mathcal{A}) = \Pr\left[\mathcal{A} \text{ succeeds in forgery}\right]$$

Proposed Scheme: Key Update

Forward Secure Pseudorandom Generator (FSPRG) [Bellare, Yee 2003]



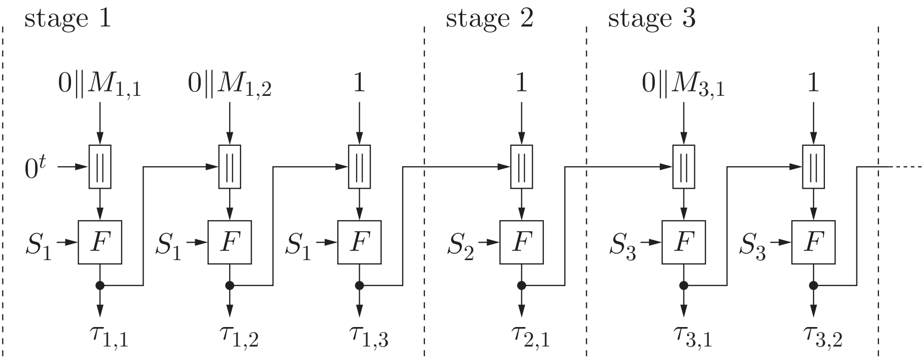
Th. Suppose that G is PRG.

$$K_1 \text{ is chosen uniformly at random}$$
$$\Downarrow$$
$$S_1 \| \cdots \| S_i \text{ looks uniformly random even if } K_{i+1} \text{ is disclosed}$$

Def. G is PRG.

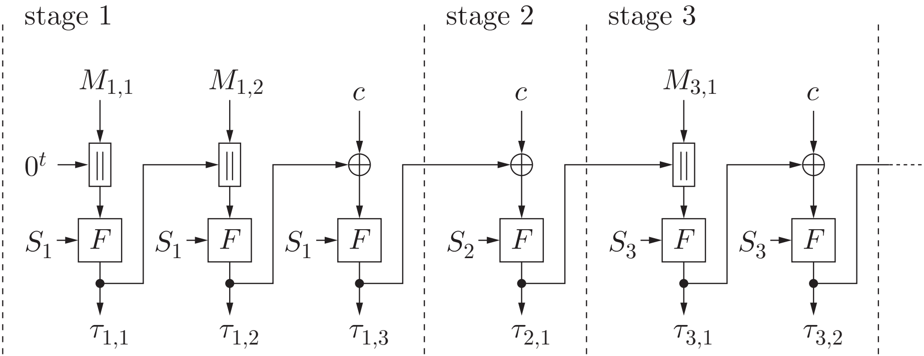
$$K_i \text{ is chosen uniformly at random} \Rightarrow K_{i+1} \| S_i \text{ looks uniformly random}$$

Proposed Scheme: Tagging



- F is a MAC function
- S_i is used for stage i
- "0^t" is the initial state
- "1" is the end marker, which prevents truncation attacks
- Tag $\tau_{i,j}$ is also used as state.

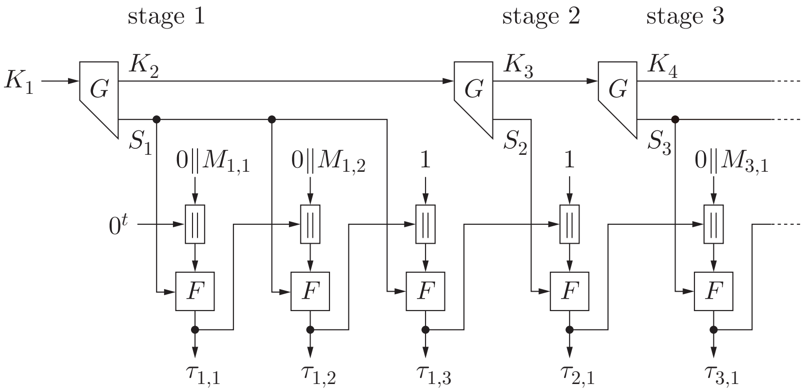
Tagging: Some Tweak



- c is a non-zero constant.

We have presented two kinds of security reductions:

- to unforgeability of F and PRG property of G
- to PRF property of F and PRG property of G



Security: Reduction to Unforgeability

Th. For any adversary $\mathcal{A}$ against $\mathsf{SAM}[F, G, n]$ with

$$\mu = (\text{no. of } \mathcal{A}\text{'s queries}) + (\text{no. of messages in } \mathcal{A}\text{'s output})$$

there exists $\mathcal{B}$ against F and $\mathcal{D}$ against G such that

$$\mathrm{Adv}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n]}(\mathcal{A}) \leq \frac{n\mu(\mu+3)}{2}\mathrm{Adv}^{\mathrm{mac}}_{F}(\mathcal{B}) + 2n \cdot \mathrm{Adv}^{\mathrm{prg}}_{G}(\mathcal{D})$$

where

- Number of $\mathcal{B}$'s queries $\leq \mu$
- Running time of $\mathcal{B} \approx$ Running time of $\mathtt{Exp}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n],\mathcal{A}}$
- Running time of $\mathcal{D} \approx$ Running time of $\mathtt{Exp}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n],\mathcal{A}}$

Security: Reduction to Indistinguishability

Th. For any adversary $\mathcal{A}$ against $\mathsf{SAM}[F, G, n]$ with

$$\mu = (\text{no. of } \mathcal{A}\text{'s queries}) + (\text{no. of messages in } \mathcal{A}\text{'s output})$$

there exists $\mathcal{C}$ against F and $\mathcal{D}$ against G such that

$$\mathrm{Adv}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n]}(\mathcal{A}) \le n \cdot \mathrm{Adv}^{\mathrm{prf}}_{F}(\mathcal{C}) + 2n \cdot \mathrm{Adv}^{\mathrm{prg}}_{G}(\mathcal{D}) + \frac{\mu^2 + \mu + 2}{2^{t+1}}$$

where

- Number of $\mathcal{C}$'s queries $\le \mu$
- Running time of $\mathcal{C} \approx$ Running time of $\mathtt{Exp}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n],\mathcal{A}}$
- Running time of $\mathcal{D} \approx$ Running time of $\mathtt{Exp}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n],\mathcal{A}}$

Comparison of the Reductions

$$\mathrm{Adv}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n]}(\mathcal{A}) \le \frac{n\mu(\mu+3)}{2}\mathrm{Adv}^{\mathrm{mac}}_{F}(\mathcal{B}) + 2n \cdot \mathrm{Adv}^{\mathrm{prg}}_{G}(\mathcal{D})$$

$$\mathrm{Adv}^{\text{fs-samac}}_{\mathsf{SAM}[F,G,n]}(\mathcal{A}) \le n \cdot \mathrm{Adv}^{\mathrm{prf}}_{F}(\mathcal{C}) + 2n \cdot \mathrm{Adv}^{\mathrm{prg}}_{G}(\mathcal{D}) + \frac{\mu^2+\mu+2}{2^{t+1}}$$

$\dfrac{n\mu(\mu+3)}{2} \gg n$, but forgery seems much more difficult than distinction:

$\mathrm{Adv}^{\mathrm{mac}}_{F}(\mathcal{B}) \ll \mathrm{Adv}^{\mathrm{prf}}_{F}(\mathcal{C}) \Leftarrow \mathcal{B}$' s power $\approx \mathcal{C}$' s power

Conclusion

Forward-Secure Sequential Aggregate Message Authentication

- Gave formalization
- Proposed a new scheme
  - with a MAC function and a PRG
  - without collision-resistant hash functions and PRPs
- Reduced the security of the scheme to
  - indistinguishability of the PRG
  - unforgeability or indistinguishability of the MAC function