On Universally Composable Time-Stamping

Peeter Laud

Tartu University & Cybernetica AS

joint work with Ahto Buldas, Märt Saarepera, Jan Willemson

Outline

- Aspects of security definitions.
- Universal Composability.
- Time-stamping.
- Universally composable time-stamping.

On definitions of secure functionalities

- Secure functionalities — for achieving some cryptographic goals.
- Examples:
  - symmetric encryption;
  - digital signatures;
  - secure channels;
  - time-stamping;
  - e-voting;
  - ...
- Security definitions exclude the undesired behaviours.

Classical definitions

- Consisted of
  - a list of undesirable scenarios (formalized);
  - statement that for all adversaries, they should occur with negligible probability.
- Hence:
  - a novel usage of the primitive means
  - new potential threat(s) means
  - a new definition.
- Security definition — a moving target.

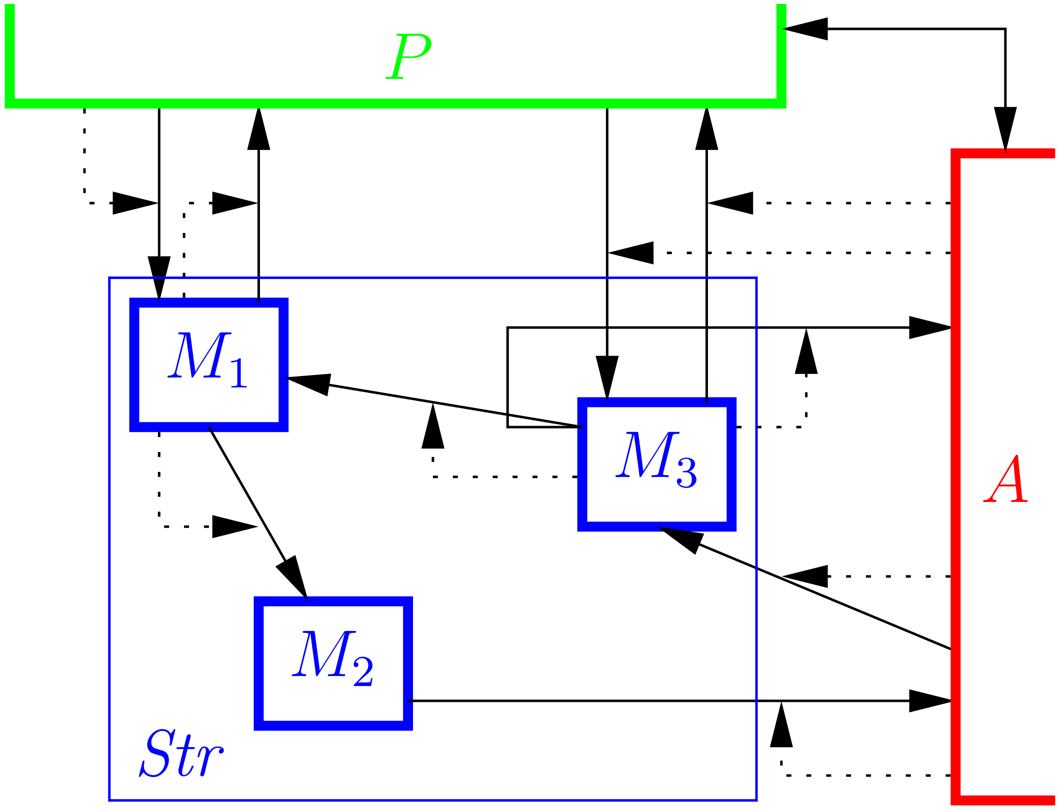
Def.s as indistinguishability of realizations

- State that the functionality must be indistinguishable from some ideal functionality.
- Ideal functionality is "obviously secure".
- Example: ideal functionality for symmetric encryption:
  - Initialization — generate a key k.
  - Encryption of x — compute $y \leftarrow \mathcal{E}_k(\mathbf{0}^{|x|})$, record (x, y), return y.
  - Decryption of y — if pair (x, y) recorded then return x else return $\mathcal{D}_k(y)$.
- Universal composability — "indistinguishability in all contexts".

Interfaces of functionalities

- A system is a set of structures.
- A structure is a collection of interacting Turing machines (ITMs).
- There are named channels between machines.
  - Machines have input and output ports.
  - Channels have buffers where messages wait scheduling.
  - Each channel is scheduled by a certain machine.
- Free ports make up interfaces. There are two:
  1. where the intended functionality is offered;
  2. for the adversary.

Structure, machines, channels, ports



Real and ideal structure

- Ideal structure (typically) has just one ITM.
  - . . . doing the ideal thing.
  - . . . which should in our opinion be sufficiently secure.
  - . . . and conceptually simple.
- In real structure
  - each ITM should be in a single geographical location;
    - and the task of a single ITM should be as small as possible;
  - secure and authentic channels should be used sparingly.

“securely realizes”

- A structure Str securely realizes the structure Str' if

$$\forall P \forall A \exists A' : view_{P \| Str \| A}(P) \approx view_{P \| Str' \| A'}(P)$$

  denote by $Str \geq Str'$.
- secure realization is uniform if

$$\forall A \exists A' \forall P : view_{P \| Str \| A}(P) \approx view_{P \| Str' \| A'}(P)$$

- secure realization is black-box if

$$\exists S \forall A \forall P : view_{P \| Str \| A}(P) \approx view_{P \| Str' \| (S \| A)}(P)$$

Why is this definition good?

- Composition theorem:

  If $Str_0 \geq Str'_0$ and Str_1 is composable with Str_0 or Str'_0 then $Str_1 \| Str_0 \geq Str_1 \| Str'_0$.
- "composable" — may not connect to the wrong ports.
  - Adversary's interfaces must be kept clear.
  - Port names may not be doubled.
- Also holds for universal and black-box realizability.

The time-stamping functionality

- Should define an order on time-stamped documents.
  - Clients submit documents to the time-stamping server (TTS).
  - TTS answers with time-stamps.
  - Given two documents and respective time-stamps their order of time-stamping should be determinable.
    - ... at least for time-stamps with long enough time between issuances.
  - Insertion of new time-stamps before existing ones should be impossible.
- The TTS is <u>untrusted</u>.

A round- and accumulator-based scheme

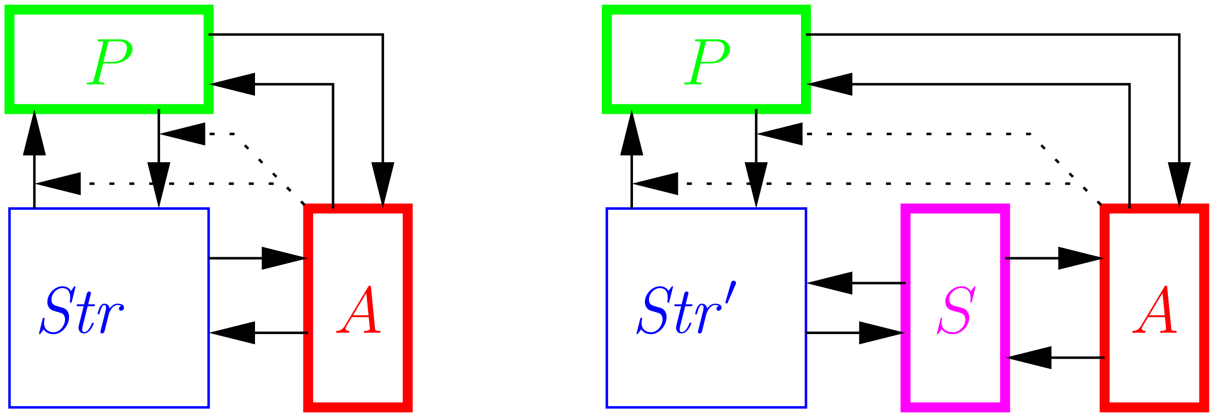
- TTS collects all time-stamped documents submitted during the current round.
- After the end of the round, it computes the accumulation of these documents and publishes it.
  - the accumulation is called the round's digest.
  - to publish $=$ to create common knowledge
- A time-stamp consists of
  - the round number;
  - proof of inclusion in the accumulation.
- Comparision: given two documents and their time-stamps:
  - compare round numbers;
  - verify proofs wrt the published digests.

Accumulators

- Accumulator: a function $D : \mathcal{P}_{\leq p(k)}(\{0,1\}^k) \longrightarrow \{0,1\}^k$.
- Also: proof-of-inclusion generator
  $P : \{0,1\}^k \times \mathcal{P}_{\leq p(k)}(\{0,1\}^k) \longrightarrow \{0,1\}^*$
  and verifier $V : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^* \longrightarrow \mathbb{B}$.
- If $x \in X$ then $V(x, D(X), P(x, X)) =$ true.
- If $x \notin X$ then very hard to find a p such that $V(x, D(X), p) =$ true.
- Example: hash trees.

On proofs of secure realizability

- Want to prove: $Str \geq Str'$ (black-box).
- Construct the simulator S.
- Show that $Str \cong (Str' \| S)$



- Done by constructing a bisimulation between the states of Str and $Str' \| S$.

On constructing the simulator

- Simulator translates between the real and the ideal world.
- Sometimes a piece of data is wanted in the ideal world that S has not yet received from the (real) A.
  - Example: S has received a ciphertext and wants to forward it to the ideal functionality Str'.
  - Str' may demand the plaintext as well.
- Then S has to substitute it with something indistinguishable.
- This indistinguishability must keep holding while the system progresses.
  - ...and more information becomes available.
- If unsubstitutable, then no simulator can be constructed.

A desired ideal functionality $\mathcal{TS}_{\mathrm{id}}$

- Take time-stamping requests (stamp, x), forward them to the adversary.
- If the adversary sends something back, forward it.
- Take end-of-round requests (EOR), forward them to the adversary.
- Take rounds' digests from the adversary.
- Take proofs-of-inclusion from the adversary, forward them to respective parties.
- Take verification requests $(\mathsf{verify}, x, c, \tau)$, where
  - x is the document;
  - c is the proof of inclusion;
  - τ is the round number.

  and answer them.

Answering verification requests

- $\mathcal{TS}_{\mathrm{id}}$ may consult with the adversary.
- If x did not exist at the end of the round τ, then the answer may not be yes.

Answering verification requests

- $\mathcal{TS}_{\mathrm{id}}$ may consult with the adversary.
- If x did not exist at the end of the round τ, then the answer may not be yes.
- What does "...did not exist..." mean???

Answering verification requests

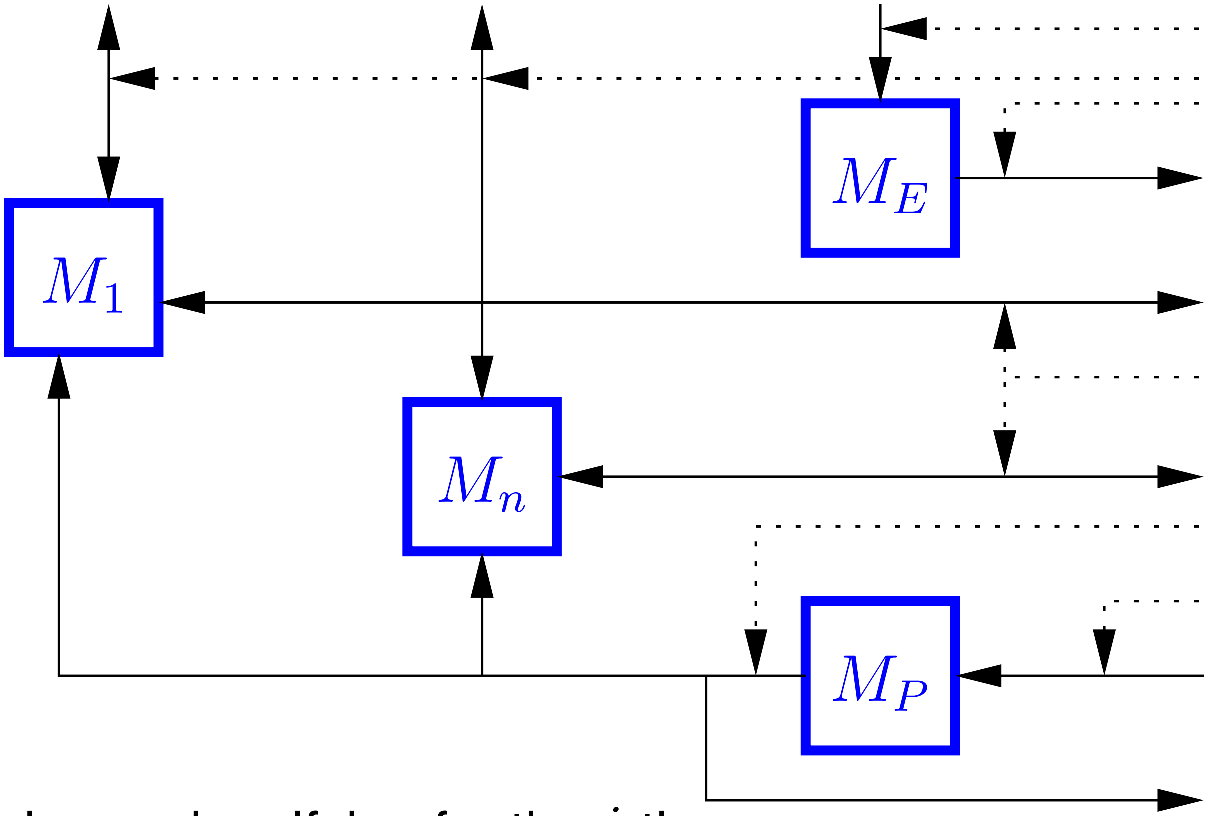
- $\mathcal{TS}_{\rm id}$ may consult with the adversary.
- If x did not exist at the end of the round τ, then the answer may not be yes.
- What does "...did not exist..." mean???
- Possible compromises:
  - With a round's digest, the adversary also returns the list of all time-stamped documents.
  - Consider only documents in time-stamping requests as time-stamped.

Timing of answering

- The pace of the computation resulting from some query in the real structure is controlled by the adversary.
  - At least when network traffic is involved.
- If the ideal structure answers this query immediately then the difference in response speeds is detectable.
- The ideal structure has to let the adversary tell it when to answer the queries.

With a round's digest, the adversary also returns the list of all time-stamped documents

The real structure



M_i — stamping and verifying for the i-th user

M_E — end-of-round signals

M_P — receives rounds' digests and publishes them

The real structure

- The machines M_i
  - forward stamping requests to the adversary;
  - forward requests for proofs-of-inclusion to the adversary and proofs-of-inclusion back to the user;
  - receive rounds' digests from M_P;
  - verify time-stamps
    - . . . independently.
- The machine M_E
  - forwards end-of-round requests to the adversary.
- The machine M_P
  - receives rounds' digests and forwards them to machines M_i.

The simulator

- The simulator has to send the list of all time-stamped documents to the ideal functionality.
  - The real functionality does not receive this list.
  - The simulator cannot make up a suitable list. . .

The simulator

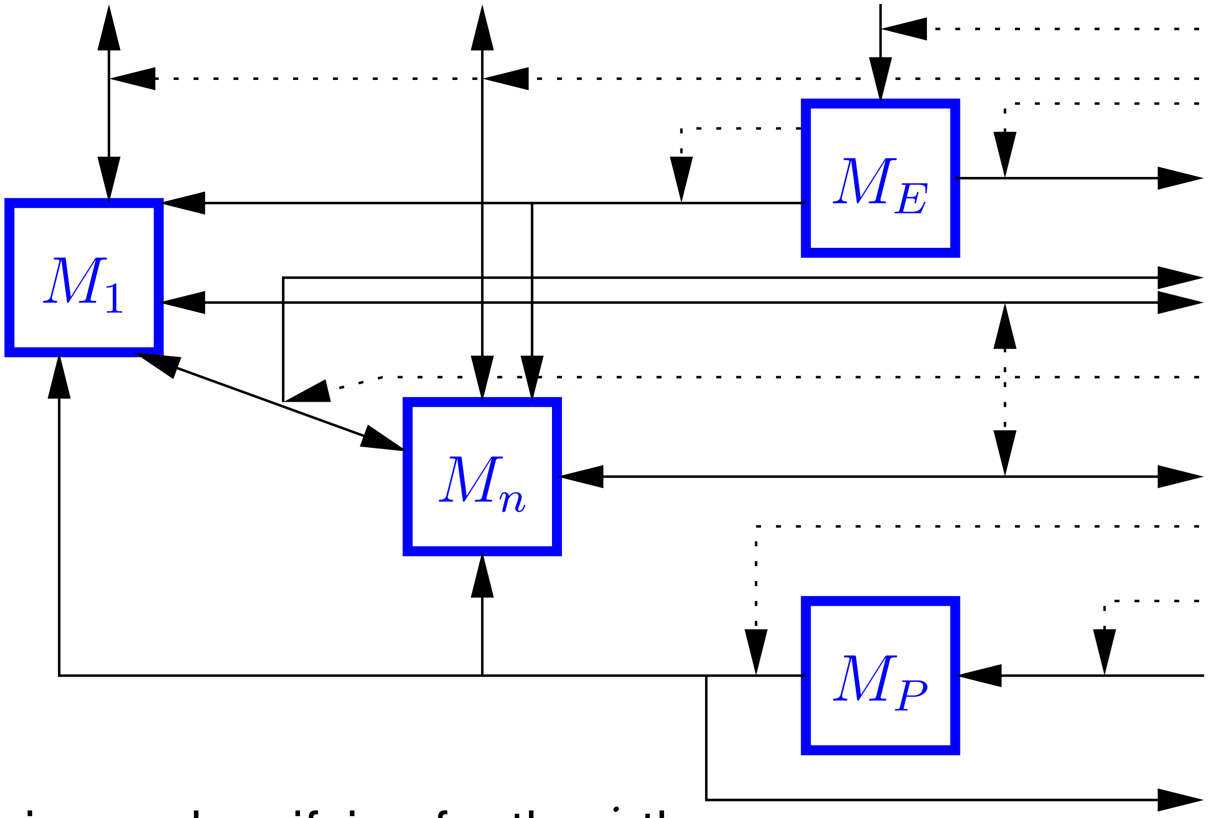
- The simulator has to send the list of all time-stamped documents to the ideal functionality.
  - The real functionality does not receive this list.
  - The simulator cannot make up a suitable list...
- The adversary has to send the list of all time-stamped documents to M_P as well.
  - "auditable time-stamping"
  - M_P checks the digest against this list.

Upside of the realization

- We can deal with corrupted parties.
  - The adversary is allowed to take over any M_i.
    - Done by sending $(\mathsf{corrupt})$ to M_i.
    - Afterwards, M_i forwards its internal state and all the received messages to the adversary and sends any message the adversary tells it.
  - The verification results for other parties are still sound.
  - Ideal structure also needs the notion of corrupted parties.

Consider only documents in time-stamping requests as time-stamped

The real structure



M_i — stamping and verifying for the i-th user

M_E — end-of-round signals

M_P — receives rounds' digests and publishes them

The real structure

- The machines M_i
  - forward stamping requests to the adversary;
  - forward requests for proofs-of-inclusion to the adversary and proofs-of-inclusion back to the user;
  - receive notes about rounds' ends from M_E;
  - receive rounds' digests from M_P;
  - verify that only the documents they time-stamped during a round are taken into account in the round's digest (as time-stamped by them);
  - send the results of this verification to other M_j-s;
  - verify time-stamps.

The real structure

- The machine M_E
  - forwards end-of-round requests to the adversary;
  - informs M_i-s about the ends of rounds.
- The machine M_P
  - receives rounds' digests and forwards them to machines M_i.

Rounds' digests

- For a round τ, there digest d_τ consists of n digests $d_\tau^{(1)}, \ldots, d_\tau^{(n)}$.
- $d_\tau^{(i)}$ is the digest of all documents submitted by the i-th user during the τ-th round.
- M_i can recalculate $d_\tau^{(i)}$ and verify that it only contains documents submitted by the i-th user.

Concluding remarks

- We have devised necessary simulators and bisimulations for showing that these real functionalities securely realize the ideal ones.
- Universal composability framework is rather restrictive for some kinds of cryptographic primitives.
- Compromises have to be made.
- Upside — composition theorem.