Secure Set Intersection with Untrusted Hardware Tokens

Thomas Schneider
Engineering Cryptographic Protocols Group, TU Darmstadt
http://encrypto.de

joint work with
• Marc Fischlin (TU Darmstadt)
• Benny Pinkas (Bar Ilan University)
• Ahmad-Reza Sadeghi (TU Darmstadt)
• Ivan Visconti (University of Salerno)

Motivation

The Best of Two Worlds

Cryptographic Protocols

- strong security + privacy guarantees
- often performance is an issue
  (e.g., Gigabytes of communication,
  heavy use of public key crypto)

+

Secure Hardware

- secure (key) storage
- trusted execution environment
- is getting cheaper

Cryptographic Protocols + Hardware ?

TECHNISCHE UNIVERSITÄT DARMSTADT

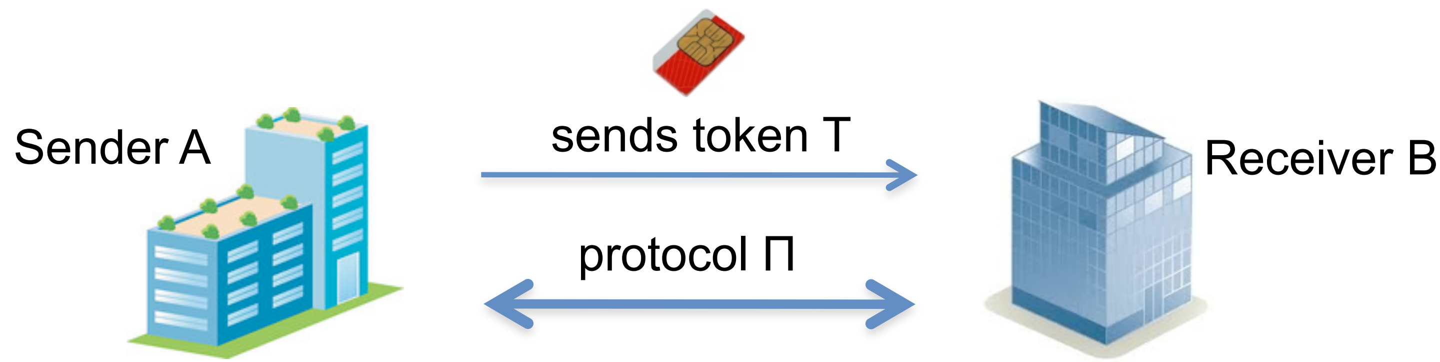
Hardware Accelerators

- allow to speed up computations
  - massive parallelism (e.g., GPU, FPGA, Cell Processor,…)

Secure Hardware

- possibilities beyond SW-only
  - secure storage
  - secure execution environment
- can be used to construct more efficient protocols
  - computation
  - communication

Background and Motivation



- Where do we use HW tokens?
  - banking, SIM cards, pay-tv, passports, health cards, …
- Why do we use HW tokens?
  - SW alone often not secure/efficient/sufficient/…
- Benefits for Practice?
  - security, efficiency, unclonability, …

Smart Cards as Secure Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT

Read-Only Memory (ROM) (> 256 KB)
- Contains operating system and applications
- Initialized during manufacturing of the device

Non-Volatile Memory (NVM) (> 128 KB)
- Read/write memory holding data after power off
- Stores user data (e.g., device serial number, application data, cryptographic secrets)
- Supports only limited number of writes (> 50.000)
- E.g., EEPROM and/or Flash

Random Access Memory (RAM) (> 25 KB)
- Read/write memory losing data after power off
- Stores temporary data during operation

Central Processing Unit (CPU)
- 8, 16 or 32 bit

Cryptographic Co-Processor
- Symmetric Encryption (typically DES, 3DES or AES)
- Cryptographic hashing (typically SHA-1)
- MAC (typically CBC-MAC)
- Public-key encryption (typically RSA/ECC)
- Signature (typically RSA/ECC)

True Random Number Generator (TRNG)
- E.g., for the generation of keys and nonces

Protection against physical attacks (e.g., against side-channel and/or invasive attacks)
- Typically compliant to FIPS 140-2
- Often certified by Common Criteria
- Includes environmental sensors (e.g., to detect voltage, frequency, temperature variations)

Counters
- E.g., for memory access control

Communication Interface
- Contact interface (1.5 – 12 MB/s)
- Contactless interface (4 – 848 KB/s)

How Secure is Secure Hardware?

Attacks [Kömmerling,Kuhn Smartcard'99]

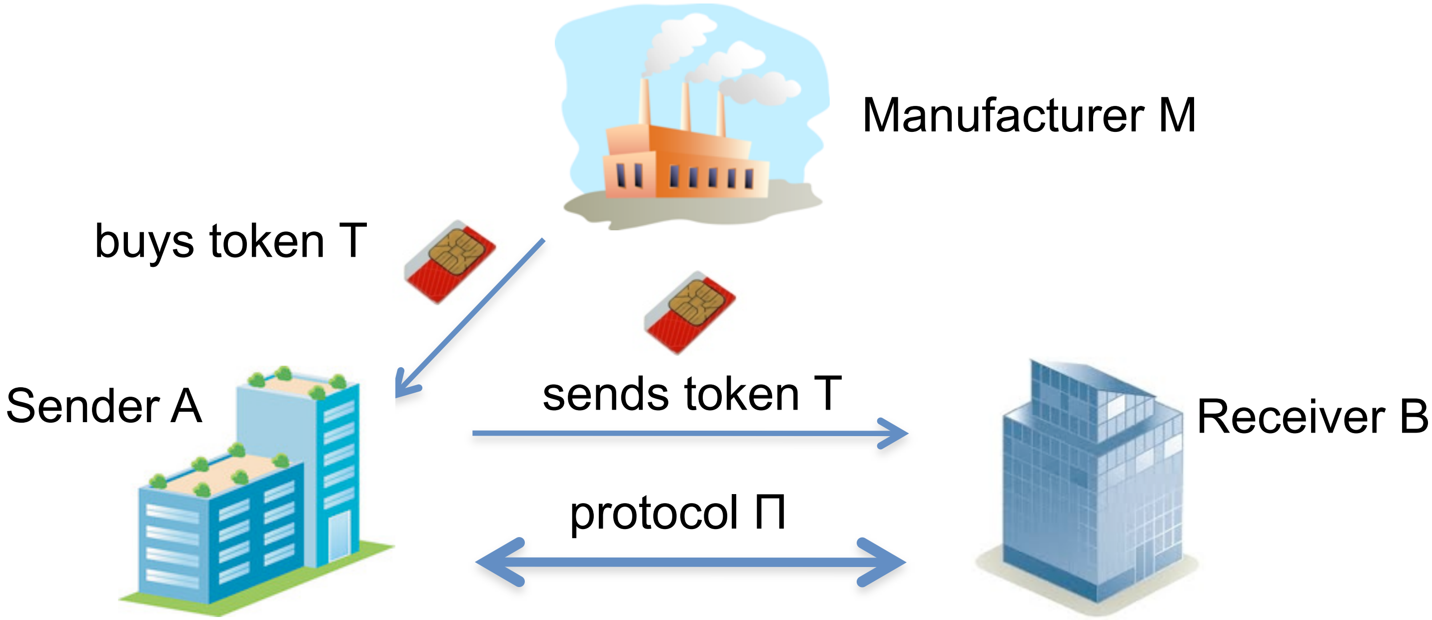
- **Micro-probing:** Obtain direct physical access to the device's memory
- **Side-channel attacks:** Analyze analog characteristics of interfaces or electromagnetic radiation of the device
- **Fault injection attacks:** Observe device's behavior under abnormal conditions (e.g., unspecified supply voltage, operating temperature, focused ion beam)
- **Example:** Hardware Attack on TPM chip [Tarnovsky BlackHat'10]

Protection Mechanisms

- Against side-channel attacks: randomized program flow, obfuscation
- Against microprobing and fault injection attacks: Tamper-detection mechanisms (e.g., temperature, voltage, frequency sensors) that erase secret data on tampering attempts

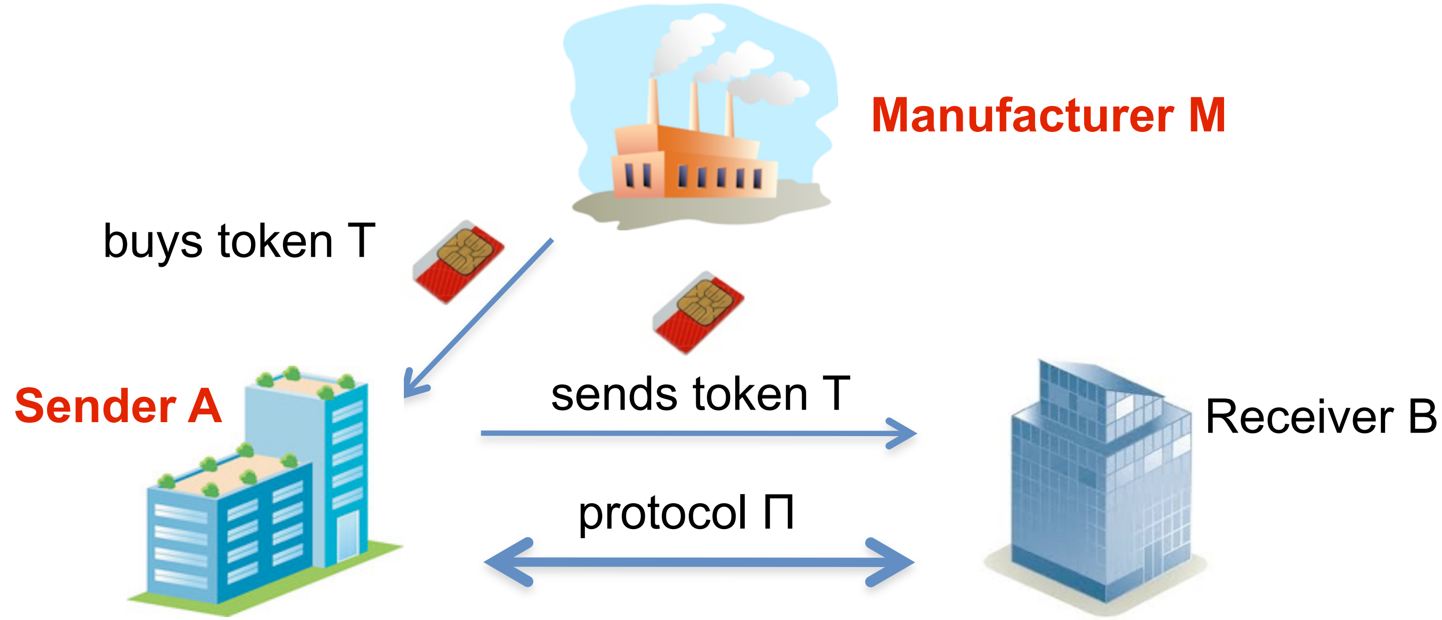
⇨ **Security of Secure Hardware is always a Trade-Off**

Goal: Tolerate Untrusted Hardware



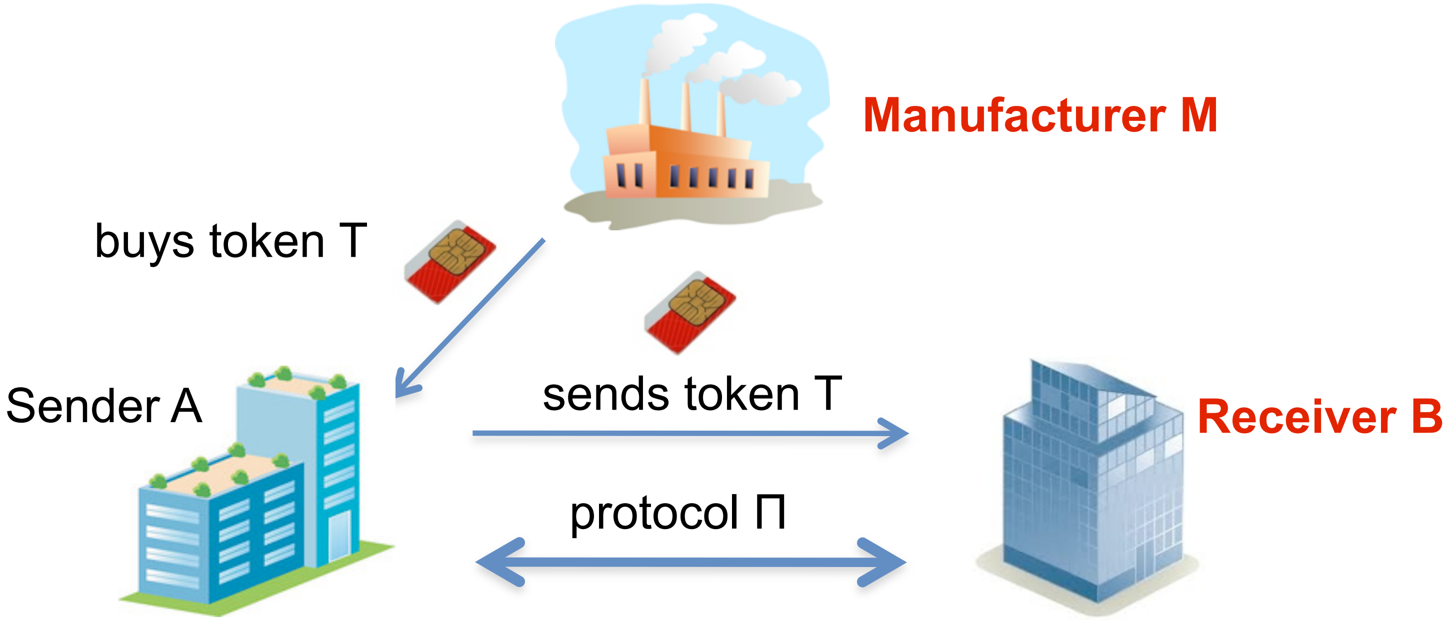
- Assumption: T is honest
  - justified if T has high level of certification (e.g. FIPS or CC)

Goal: Tolerate Untrusted Hardware



- Goal 1: B does not trust T (A could send cheating T)
  - A colludes with M
  - hardware trojans
  - bugs, …

Goal: Tolerate Untrusted Hardware

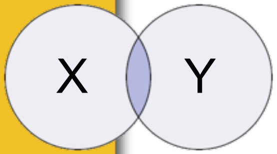


- Goal 2: A does not trust T (B could break into T)
  - B colludes with M
  - side-channel attacks
  - bugs, ...

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

SECURE SET INTERSECTION

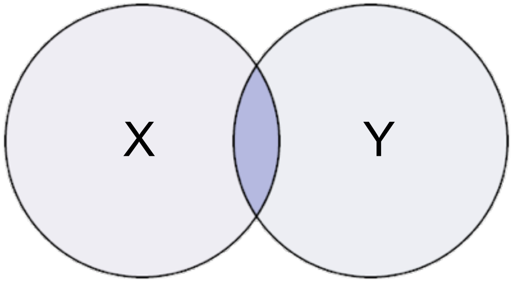


RECEIVER B DOES NOT TRUST T

SENDER A DOES NOT TRUST T

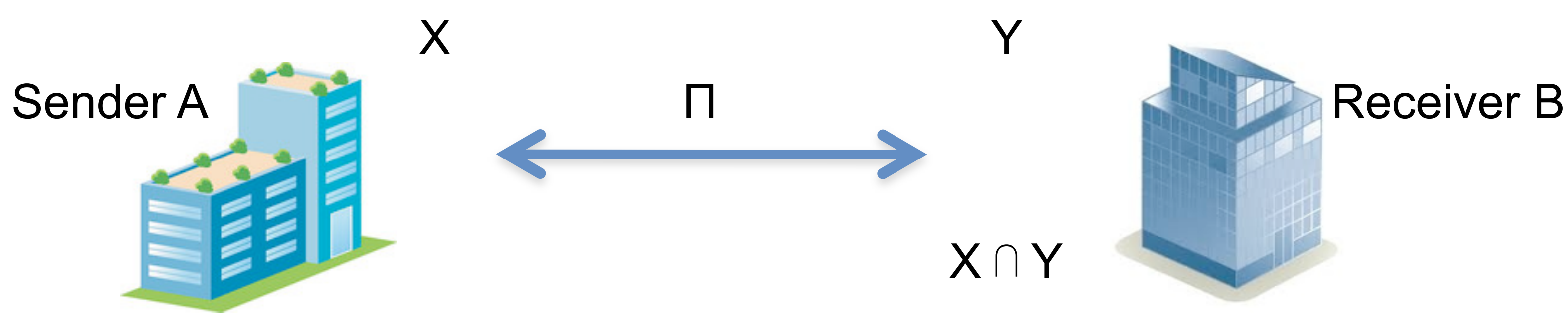
CONCLUSION

Secure Set Intersection



Secure Set Intersection

- Inputs: A has set X, B has set Y
- Output: B obtains nothing but X∩Y
- Application Examples:
  - Organizations: joint properties
  - Governments: joint criminal suspects
  - Companies: joint customers
  - People: common contacts



Set Intersection Protocols

- **SW**-only Protocols based on PK crypto
  ([Freedman,Nissim,Pinkas EUROCRYPT'04], …,
  [De Cristofaro,Kim,Tsudik ASIACRYPT'10])
  - at least O(|X|+|Y|) PK operations
  - at least O(|X|+|Y|) communication

- **SW**-only Protocol of [Huang,Evans,Katz NDSS'12]:
  evaluate garbled circuit with $|C| \sim \sigma N \log N$ gates
  - O(|C|) symmetric crypto (Hash function)
  - O(|C|) communication
  - Better performance than previous PK-based protocols

σ: bit length of elements
$N = |X| + |Y|$

Set Intersection Protocols

- **HW**-based Protocol of [Hazay,Lindell CCS'08]:
  - O(|X|+|Y|) symmetric crypto (fixed key AES)
  - O(|X|) communication
  - Token T
    - **Trusted by both players**
    - Constant amount of memory
      (using O(|X|+|Y|) memory T could simply compute the intersection itself)

Set Intersection Protocol of [HL08] (simplified)

TECHNISCHE UNIVERSITÄT DARMSTADT

F: PRP, D={0,1}^{t}
t: symm. sec. param
(e.g., F=AES, t=128)

$\mathcal{A}$: $X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$

$\mathcal{B}$: $Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$

Setup Phase: $k, \mathrm{OK} \in_R D$

init $\mathcal{T}$: $k, \mathrm{OK}, n_\mathcal{B}$ $\xrightarrow{\mathcal{T}}$ $\mathcal{T}$

T will answer at most n_B queries

Online Phase:

$\mathcal{B}$: $\forall y_j \in Y$: $\xrightarrow{y_j}$ $\mathcal{T}$: $\bar{y}_j = F_k(y_j)$; $\xleftarrow{\bar{y}_j}$

$\mathcal{B}$ $\xrightarrow{\text{done}}$ $\mathcal{T}$: invalidate k, $\mathrm{OK}' = \mathrm{OK}$; $\xleftarrow{\mathrm{OK}'}$

$\mathcal{B}$: $\mathrm{OK}'' = \mathrm{OK}'$ $\xrightarrow{\mathrm{OK}''}$ $\mathcal{A}$: $\mathrm{OK}'' \stackrel{?}{=} \mathrm{OK}$

$\mathcal{A}$: $\bar{X} = \{F_k(x)\}_{x \in X}$ $\xrightarrow{\bar{X}}$ $\mathcal{B}$: $X \cap Y = \{y_j | \bar{y}_j \in \bar{X}\}$

- Efficiency: Runtime on Java Card ≈ 50ms · |Y|
- Security: Universal Composability (UC)
- Assumption: T trusted by both parties

  (use simulator's control over T to extract inputs of A,B)

EC SPRIDE
EUROPEAN CENTER FOR SECURITY AND PRIVACY BY DESIGN

Our Contribution

- Efficient Secure Set Intersection with Untrusted HW tokens:
  - Use symmetric crypto only
  - Token(s) not trusted by both parties

fully trusted

untrusted by B

untrusted by A

- Security:

UC

Input Privacy w.r.t. Malicious Adversaries
Output Correctness w.r.t. Covert Adversaries

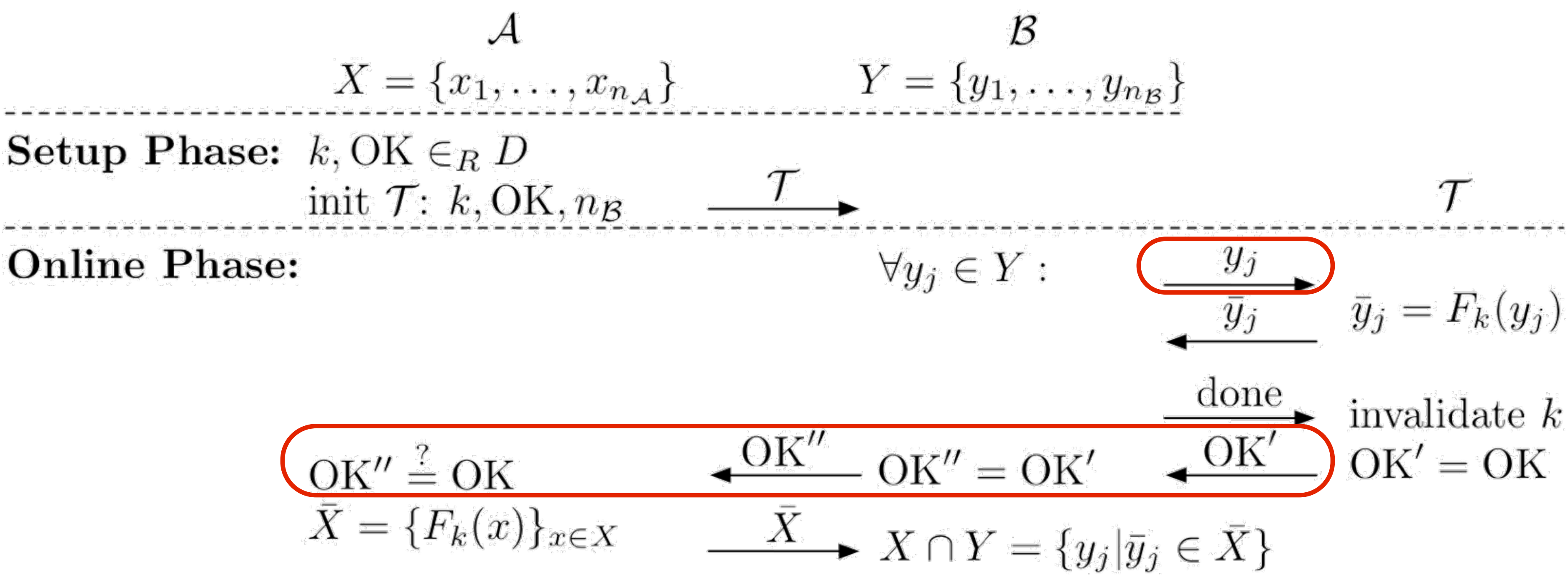
Fall-Back Security

Any Cheating Attempts
- can breach only correctness
- but not privacy
- detected with high probability (e.g., ε=1/2)

Goal 1: Receiver B does not trust T

B does not trust T: Protect B's Privacy (1. try)

- Problem 1: T sees B's input, so malicious T could use covert channel inside OK message to send information back to A
- => reorder messages to eliminate OK message



B does not trust T: Protect B's <u>Privacy</u> (2. try)

TECHNISCHE UNIVERSITÄT DARMSTADT

- another Problem: unable to simulate against malicious B that could change his inputs y_j adaptively
- => add another layer of encryption

$\mathcal{A}$ | $\mathcal{B}$

$X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$ | $Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$

Setup Phase:

$k, \mathrm{OK} \in_R D$

init $\mathcal{T}$: $k, \mathrm{OK}, n_\mathcal{B}$ $\xrightarrow{\mathcal{T}}$ $\mathcal{T}$

Online Phase:

$\bar{X} = \{F_k(x)\}_{x \in X}$ $\xrightarrow{\bar{X}}$ $\forall y_j \in Y:$ $\xrightarrow{y_j}$ $\xleftarrow{\bar{y}_j}$ $\bar{y}_j = F_k(y_j)$

$X \cap Y = \{y_j | \bar{y}_j \in \bar{X}\}$

EC SPRIDE EUROPEAN CENTER FOR SECURITY AND PRIVACY BY DESIGN

B does not trust T: Protect B's Privacy

F: PRP, f=PRF, D={0,1}^{t}
t: symm. sec. param
(e.g., F=f=AES, t=128)

$\mathcal{A}$: $X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$

$\mathcal{B}$: $Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$

Setup Phase:

$k, s \in_R D$

init $\mathcal{T}$: $k, s, n_\mathcal{B}$ $\xrightarrow{\mathcal{T}}$ $\mathcal{T}$

Online Phase:

$\bar{X} = \{F_k(x)\}_{x \in X}$ $\xrightarrow{\bar{X}}$ $\forall j \in \{1, .., n_\mathcal{B}\}$:

$\xrightarrow{y_j}$ $p_j = f_s(j)$

$\xleftarrow{\bar{y}'_j}$ $\bar{y}'_j = F_k(y_j) \oplus p_j$

afterwards

$\xrightarrow{\text{done}}$ invalidate k

$\xleftarrow{p_j}$ $p_j = f_s(j)$

$\bar{y}_j = \bar{y}'_j \oplus p_j$

$X \cap Y = \{y_j | \bar{y}_j \in \bar{X}\}$

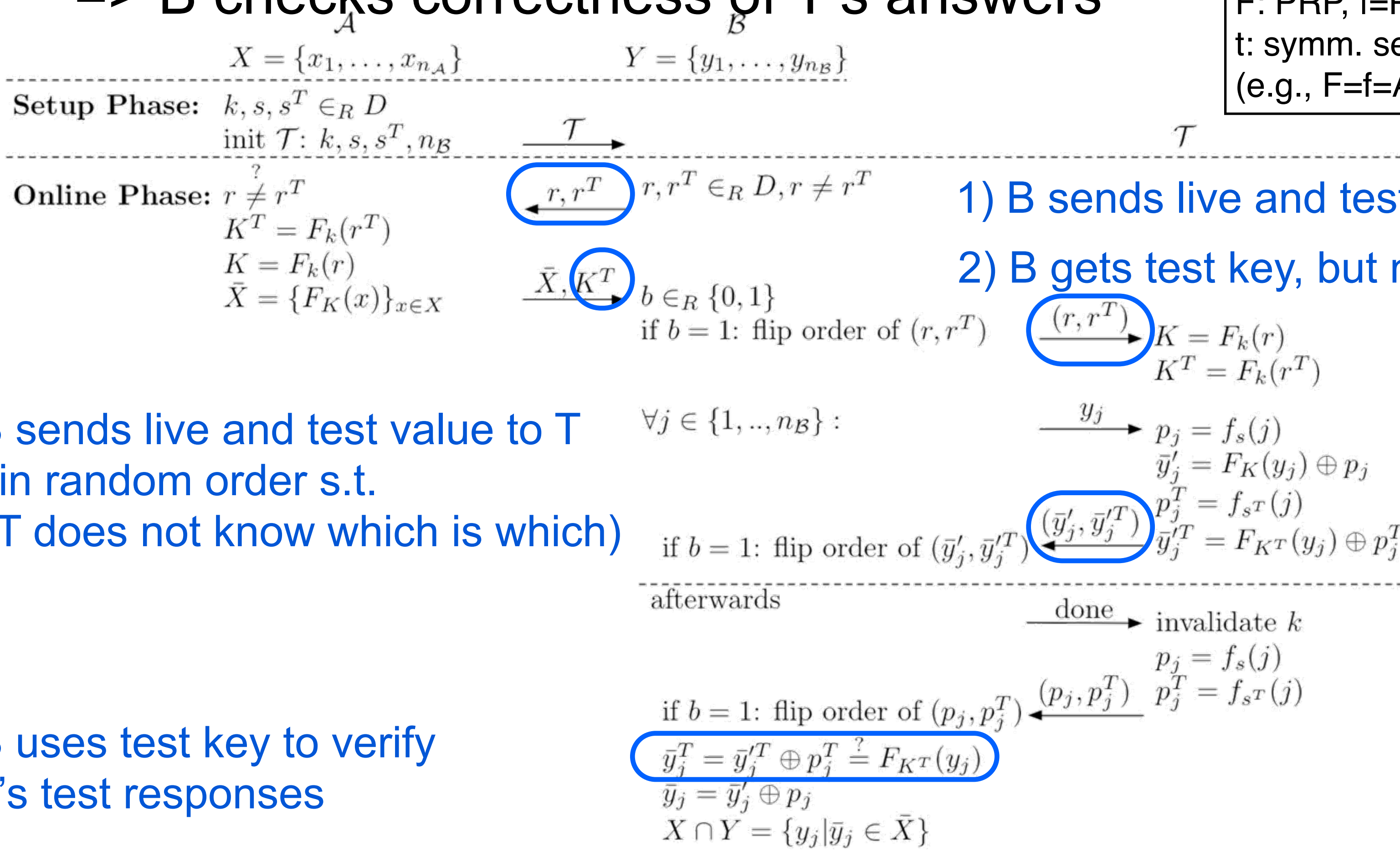
- workload of T increased by factor of 3

B does not trust T: Protect B's Correctness

- Idea (adapted from [Kolesnikov TCC'10]): use live + test run

  => B checks correctness of T's answers

F: PRP, f=PRF, D={0,1}^t
t: symm. sec. param
(e.g., F=f=AES, t=128)



1) B sends live and test value to A

2) B gets test key, but not live key

3) B sends live and test value to T
(in random order s.t.
T does not know which is which)

4) B uses test key to verify
T's test responses

Goal 2: Sender A does not trust T

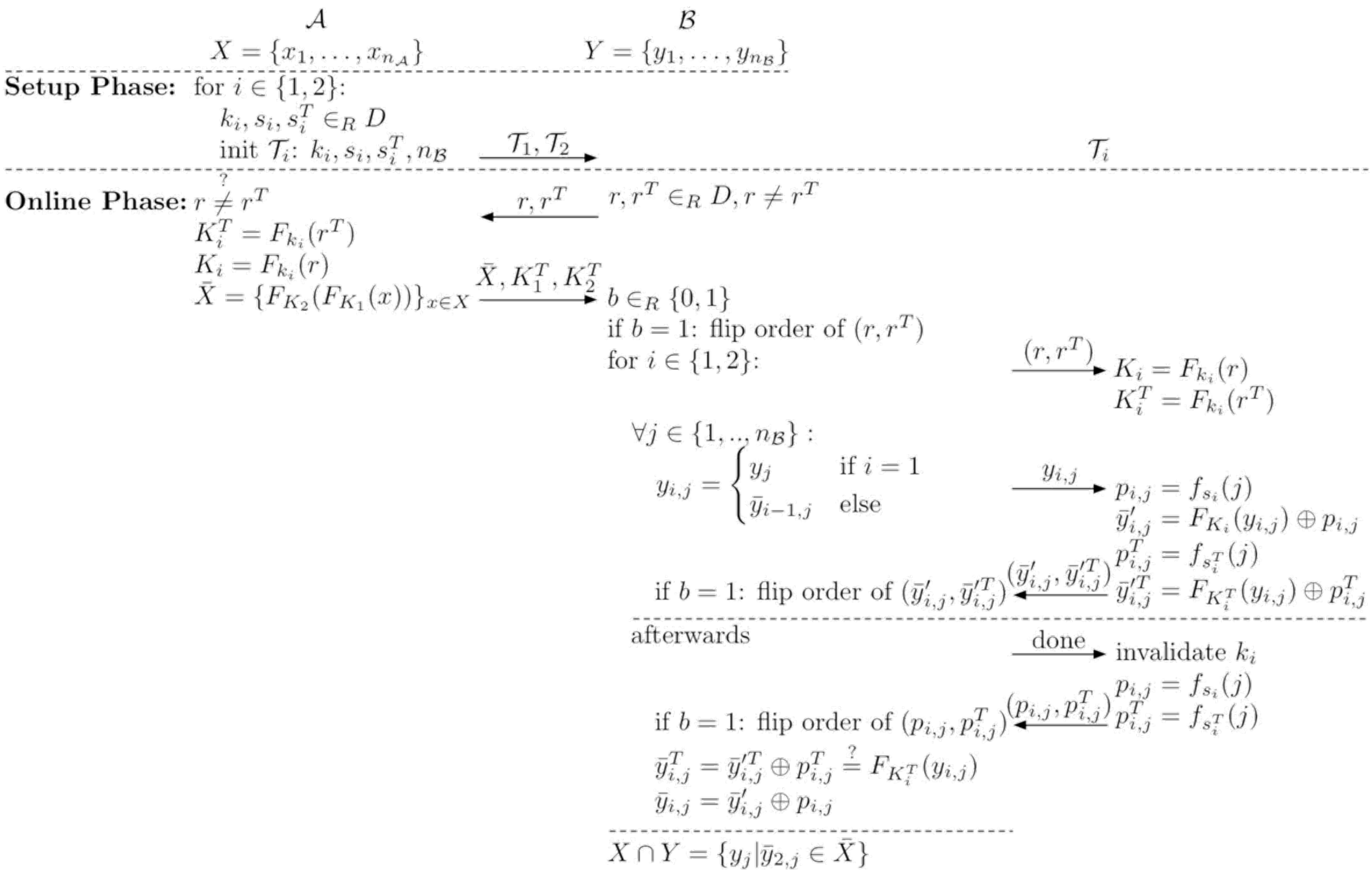
A does not trust T: Protect A's Privacy

- If B can break into T to learn its entire state,
  A has no advantage in using T any more
- Our approach:
  - Use multiple Tokens T_i (from different manufacturers M_i)
  - Assume that receiver B can break into all but one token
- Idea:
  - Use sequential composition of PRPs F_{Ki}

F_{K1} F_{K2} F_{K3}

A does not trust T: Protect A's Privacy

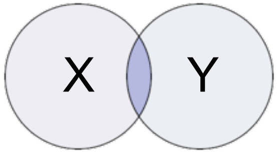
- Use Multiple Tokens Sequentially



Conclusion

Summary

- Efficient Extensions for Token-based Set Intersection [Hazay,Lindell CCS'08]:



  - Similar communication: ~ |X|
  - Tolerate malicious token sent by A
    - Workload by T increases by factor 3

  - Tolerate B breaking all but one of n tokens
    - Workload by A,B increases linearly in n

  - Security:
    - Privacy against malicious adversaries
    - Correctness against covert adversaries
    - Fall-back UC security if tokens are trusted

Secure Set Intersection with Untrusted Hardware Tokens

Questions ?