Simple, Efficient and Strongly KI-Secure Hierarchical Key Assignment Schemes

Eduarda S. V. Freire and Bertram Poettering and Kenny G. Paterson

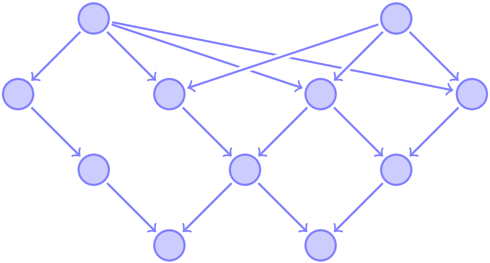
Information Security Group
Royal Holloway, University of London

bertram.poettering@rhul.ac.uk

CT-RSA, February 27, 2013

Information Security Group

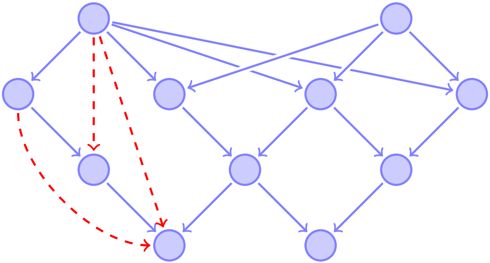
Hierarchical Key Assignment Scheme (KAS)



Setting and terminology

- topology: poset $(V, \leq)$ (reflexive, antisymmetric, transitive)
- representation by transitive reduction, called access graph
- nodes $u \in V$ called (security) class
  - individual private information S_u
  - individual (encryption) key k_u
- functionality: given S_u, compute k_v for any $v \leq u$

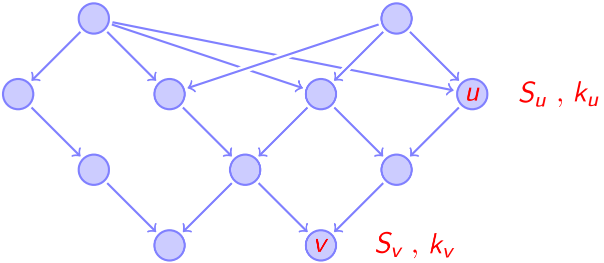
Hierarchical Key Assignment Scheme (KAS)



Setting and terminology

- topology: poset $(V, \leq)$ (reflexive, antisymmetric, transitive)
- representation by transitive reduction, called access graph
- nodes $u \in V$ called (security) class
  - individual private information S_u
  - individual (encryption) key k_u
- functionality: given S_u, compute k_v for any $v \leq u$

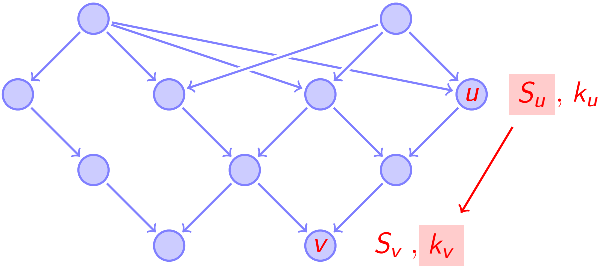
Hierarchical Key Assignment Scheme (KAS)



Setting and terminology

- topology: poset $(V, \leq)$ (reflexive, antisymmetric, transitive)
- representation by transitive reduction, called access graph
- nodes $u \in V$ called (security) class
  - individual private information S_u
  - individual (encryption) key k_u
- functionality: given S_u, compute k_v for any $v \leq u$

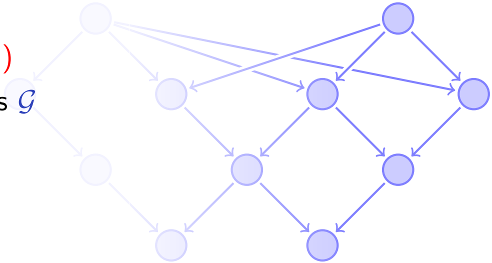
Hierarchical Key Assignment Scheme (KAS)



Setting and terminology

- topology: poset $(V, \leq)$ (reflexive, antisymmetric, transitive)
- representation by transitive reduction, called access graph
- nodes $u \in V$ called (security) class
  - individual private information S_u
  - individual (encryption) key k_u
- functionality: given S_u, compute k_v for any $v \leq u$

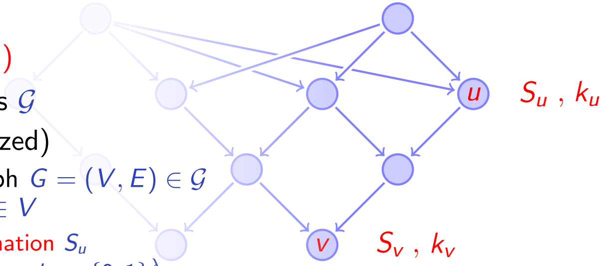
Syntax and Functionality of KAS



Syntax of KAS (simplified)

- class of access graphs $\mathcal{G}$

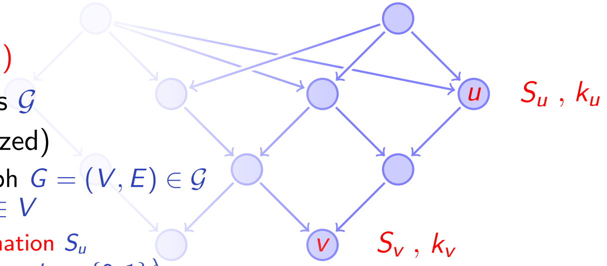
Syntax and Functionality of KAS



Syntax of KAS (simplified)

- class of access graphs $\mathcal{G}$
- $\mathsf{Gen}(1^\lambda, G)$ (randomized)
  - input: access graph $G = (V, E) \in \mathcal{G}$
  - output: for all $u \in V$
    - private information S_u
    - (encryption) key $k_u \in \{0,1\}^\lambda$

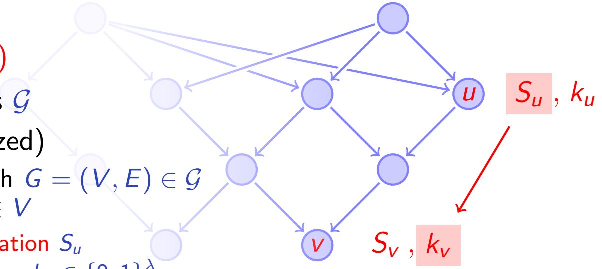
Syntax and Functionality of KAS



Syntax of KAS (simplified)

- class of access graphs $\mathcal{G}$
- $\mathsf{Gen}(1^\lambda, G)$ (randomized)
  - input: access graph $G = (V, E) \in \mathcal{G}$
  - output: for all $u \in V$
    - private information S_u
    - (encryption) key $k_u \in \{0,1\}^\lambda$
- $\mathsf{Derive}(S_u, v)$ (deterministic)
  - input: private information S_u and class $v \leq u$
  - output: key $k \in \{0,1\}^\lambda$

Syntax and Functionality of KAS



Syntax of KAS (simplified)

- class of access graphs $\mathcal{G}$
- $\mathrm{Gen}(1^\lambda, G)$ (randomized)
  - input: access graph $G = (V, E) \in \mathcal{G}$
  - output: for all $u \in V$
    - private information S_u
    - (encryption) key $k_u \in \{0,1\}^\lambda$
- $\mathrm{Derive}(S_u, v)$ (deterministic)
  - input: private information S_u and class $v \leq u$
  - output: key $k \in \{0,1\}^\lambda$

Correctness requirement

we require $\mathrm{Derive}(S_u, v) = k_v$ for all u and $v \leq u$.

Applications of Key Assignment Schemes

Applications of KAS

- access control
  - example: patient records in hospital
    - limited access for nurses
    - access rights of doctors in dependence of seniority
  - example: sensors in building management
    - employees can access local light and temperature sensors
    - managers can access all sensors installed on given floor
    - facility manager can access smoke sensors and intrusion detection on all floors
    - firefighters can access smoke sensors on all floors
- database security
- content distribution and digital broadcasting
- military/government communication

Applications of Key Assignment Schemes

Applications of KAS

- access control
  - example: patient records in hospital
    - limited access for nurses
    - access rights of doctors in dependence of seniority
  - example: sensors in building management
    - employees can access local light and temperature sensors
    - managers can access all sensors installed on given floor
    - facility manager can access smoke sensors and intrusion detection on all floors
    - firefighters can access smoke sensors on all floors
- database security
- content distribution and digital broadcasting
- military/government communication

Variants of KAS

- time-dependent constraints
- dynamic addition or removal of classes
- revocation handling

History of Key Assignment Schemes

Important achievements in key assignment (heavily biased excerpt)

- [AklTay83]
  - idea to implement access control through key assignment
  - RSA-based construction (w/o proof)
- [AtaBlaFazFri05]
  - first formal security model (KR+KI)
  - PRF-based construction
- [CraMarWil06]
  - overview paper, classifying 27 schemes into 5 design categories
- [D'ArSanFerMas10]
  - formal analysis of Akl-Taylor scheme
  - RSA vs. strong RSA?
- [CraDauMar10]
  - idea of chain partition method (w/o proof)

Contributions of this work

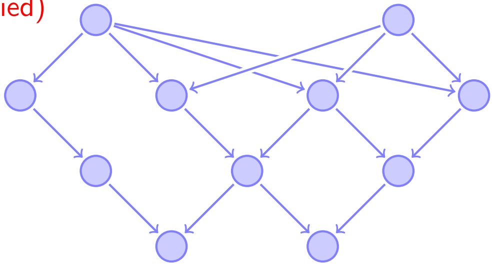
Our contributions

- new security model
  - we claim: security definitions published so far are unrealistic
  - we introduce new formal security model, fixing all problems
  - we give counterexample to separate new from old models
- security analysis of chain-based construction
  - we prove generic security of the chain partition method
- new constructions of KAS
  - we construct highly efficient KAS, based on PRFs, PRGs, ...
  - we establish formal security via reductionist proofs
- assessment of practicality
  - we propose parameters for concrete instantiations of our KAS
    - based on HMAC, AES, BBS, others
  - we compare our schemes with other published KAS
  - we discuss possible efficiency tradeoffs

Security notions of KAS: previous models

Previous security definitions (unified)

$\mathsf{Exp}^{\mathsf{KI\text{-}ST},b}_{\mathcal{A},G}(1^\lambda)$

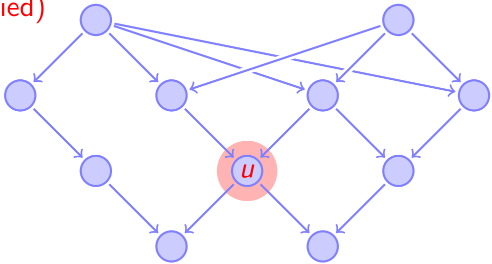


Security notions of KAS: previous models

Previous security definitions (unified)

$\mathsf{Exp}^{\mathsf{KI\text{-}ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$

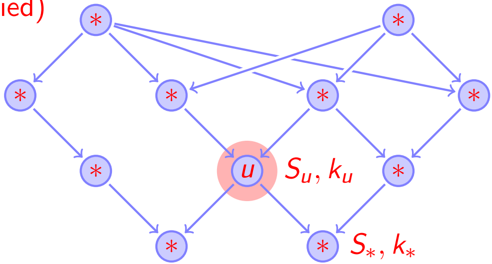


Security notions of KAS: previous models

Previous security definitions (unified)

$\mathrm{Exp}^{\text{KI-ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$

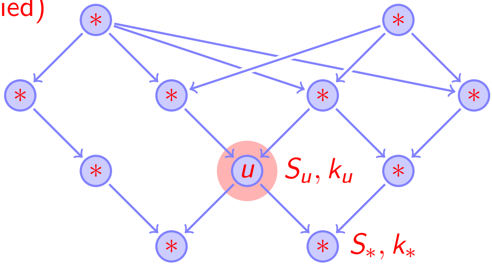


Security notions of KAS: previous models

Previous security definitions (unified)

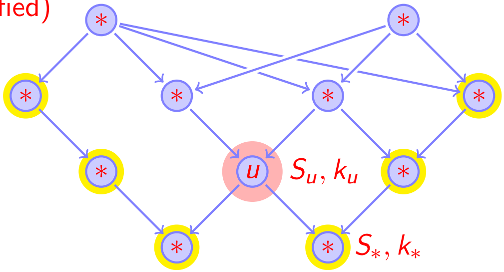
$\mathsf{Exp}^{\mathsf{KI\text{-}ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathsf{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$



Security notions of KAS: previous models

Previous security definitions (unified)

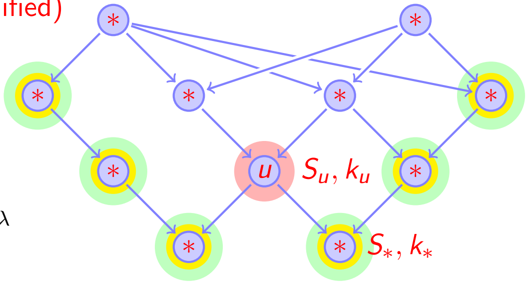


$\mathrm{Exp}_{\mathcal{A},G}^{\text{KI-ST},b}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, T)$
    - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u

Security notions of KAS: previous models

Previous security definitions (unified)

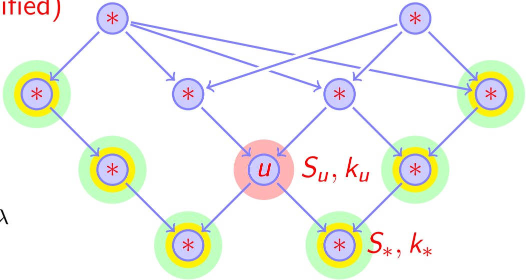


$\mathrm{Exp}_{\mathcal{A},G}^{\text{KI-ST},b}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, T)$
  - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
  - $\mathcal{A}$ can compute $\vec{k}_{u \not\leq v} = \{k_v : u \not\leq v\}$

Security notions of KAS: previous models

Previous security definitions (unified)

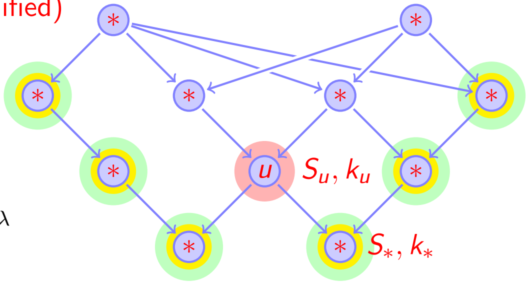


$\mathrm{Exp}^{\text{KI-ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, T)$
  - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
  - $\mathcal{A}$ can compute $\vec{k}_{u \not\leq v} = \{k_v : u \not\leq v\}$
- return d

Security notions of KAS: previous models

Previous security definitions (unified)



$\mathrm{Exp}^{\text{KI-ST},b}_{\mathcal{A},G}(1^\lambda)$

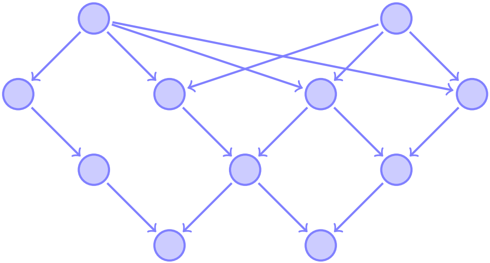
- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, T)$
  - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
  - $\mathcal{A}$ can compute $\vec{k}_{u \not\leq v} = \{k_v : u \not\leq v\}$
- return d

Variants

- key indistinguishability vs. key recoverability
- static vs. dynamic adversaries

Security notions of KAS: revised model

Updated security model

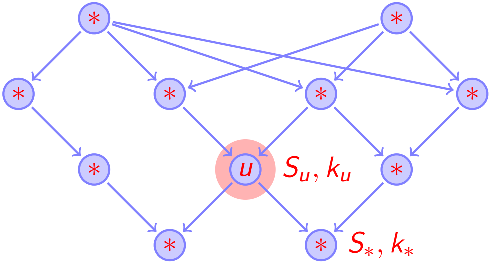


Security notions of KAS: revised model

Updated security model

$\mathsf{Exp}_{\mathcal{A},G}^{\mathsf{S\text{-}KI\text{-}ST},b}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathsf{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$

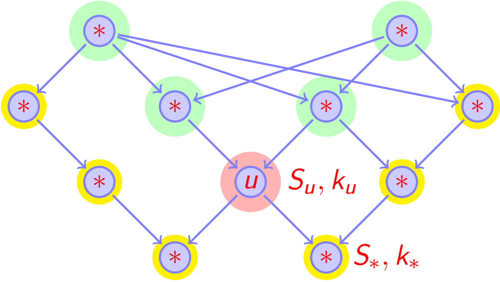


Security notions of KAS: revised model

Updated security model

$\mathrm{Exp}^{\text{S-KI-ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, \vec{k}_{u<v}, T)$
    - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
    - $\vec{k}_{u<v} = \{k_v : u < v\}$ keys 'superior' to u

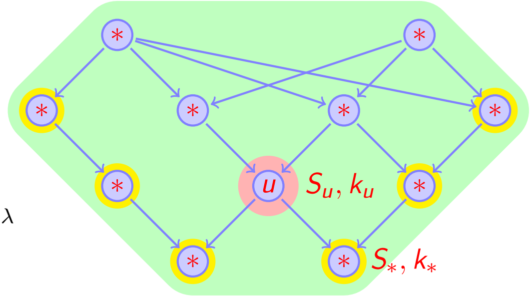


Security notions of KAS: revised model

Updated security model

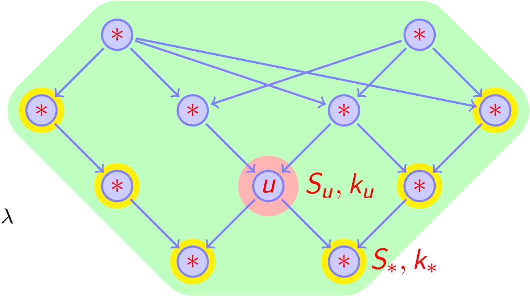
$\mathrm{Exp}_{\mathcal{A},G}^{\text{S-KI-ST},b}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, \vec{k}_{u<v}, T)$
    - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
    - $\vec{k}_{u<v} = \{k_v : u < v\}$ keys 'superior' to u
    - $\mathcal{A}$ can compute $\vec{k}_{u \neq v} = \{k_v : u \neq v\}$



Security notions of KAS: revised model

Updated security model

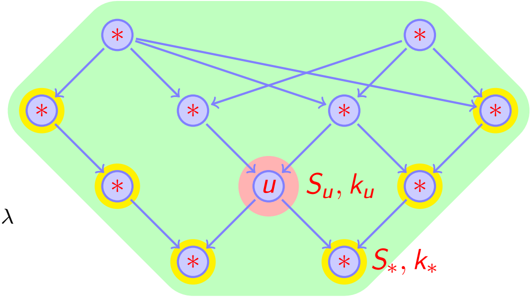


$\mathrm{Exp}^{\text{S-KI-ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, \vec{k}_{u<v}, T)$
  - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
  - $\vec{k}_{u<v} = \{k_v : u < v\}$ keys 'superior' to u
  - $\mathcal{A}$ can compute $\vec{k}_{u \neq v} = \{k_v : u \neq v\}$
- return d

Security notions of KAS: revised model

Updated security model



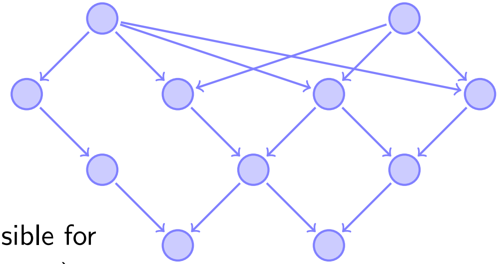
$\mathrm{Exp}^{\text{S-KI-ST},b}_{\mathcal{A},G}(1^\lambda)$

- $u \leftarrow \mathcal{A}(G)$
- $(\vec{S}, \vec{k}) \leftarrow \mathrm{Gen}(1^\lambda, G)$
- if $b = 1$ then $T \leftarrow k_u$
- if $b = 0$ then $T \leftarrow_R \{0,1\}^\lambda$
- $d \leftarrow \mathcal{A}(\vec{S}_{u \not\leq v}, \vec{k}_{u<v}, T)$
  - $\vec{S}_{u \not\leq v} = \{S_v : u \not\leq v\}$ secret information not 'superior' to u
  - $\vec{k}_{u<v} = \{k_v : u < v\}$ keys 'superior' to u
  - $\mathcal{A}$ can compute $\vec{k}_{u \neq v} = \{k_v : u \neq v\}$
- return d

Comparing the models

- Claim: all previous security models useless in practice
- we give counterexample to formally separate the models

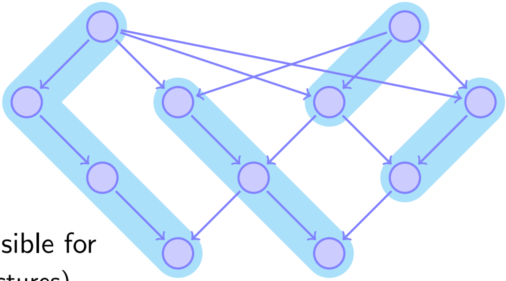
Constructing key assignment: from chains to posets



Challenge

- KAS constructions seem feasible for
  - chains (linear access structures)
  - trees (strict hierarchies)
- but constructions for general posets?

Constructing key assignment: from chains to posets



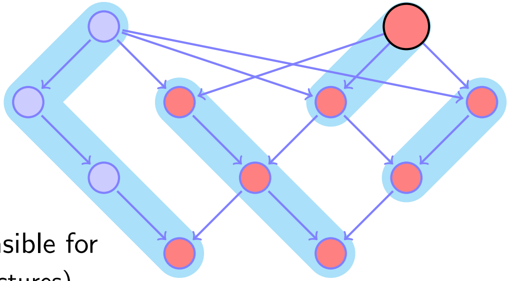
Challenge

- KAS constructions seem feasible for
  - chains (linear access structures)
  - trees (strict hierarchies)
- but constructions for general posets?

Chain partition method ([CraDauMar10], w/o proof)

- cover poset with disjoint key assignment chains

Constructing key assignment: from chains to posets



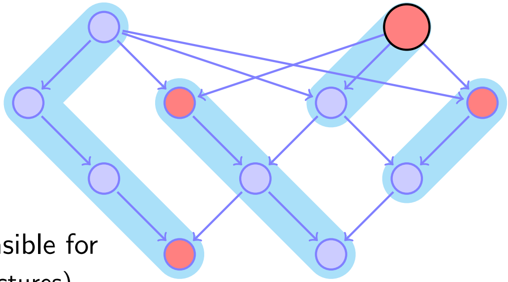
Challenge

- KAS constructions seem feasible for
  - chains (linear access structures)
  - trees (strict hierarchies)
- but constructions for general posets?

Chain partition method ([CraDauMar10], w/o proof)

- cover poset with disjoint key assignment chains
- for each class u, intersection of chains with $\{v : v \leq u\}$ is suffix

Constructing key assignment: from chains to posets



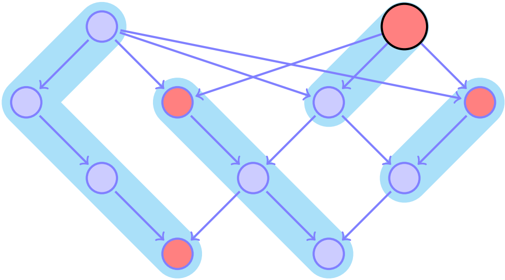
Challenge

- KAS constructions seem feasible for
  - chains (linear access structures)
  - trees (strict hierarchies)
- but constructions for general posets?

Chain partition method ([CraDauMar10], w/o proof)

- cover poset with disjoint key assignment chains
- for each class u, intersection of chains with $\{v : v \leq u\}$ is suffix
- each class stores (at most) one entry per chain

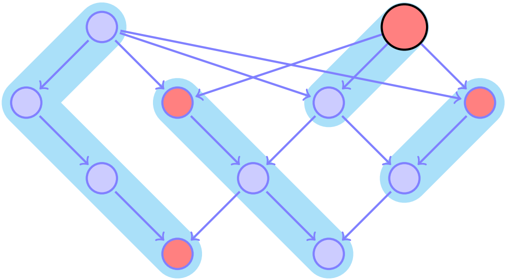
Constructing key assignment: from chains to posets



Dilworth's theorem (1950)

> *In finite posets, the minimum number of chains in any partition into chains equals the maximum cardinality of any antichain. We call this number the width of the poset.*

Constructing key assignment: from chains to posets



Dilworth's theorem (1950)

In finite posets, the minimum number of chains in any partition into chains equals the maximum cardinality of any antichain. We call this number the width of the poset.

Theorem

The chain partition method provides S-KI-ST *security, assuming all KAS chains are* S-KI-ST *secure.*

PRF-based KAS construction for chains

Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

PRF-based KAS construction for chains

Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathrm{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and 'k', 'S' $\in D$
  - instantiations: HMAC, BBS+GGM

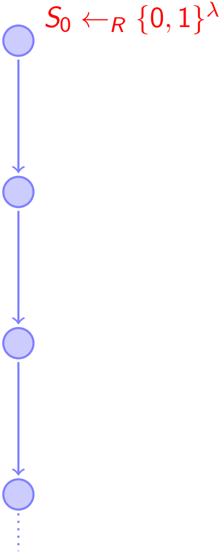
PRF-based KAS construction for chains

Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathrm{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and 'k', 'S' $\in D$
  - instantiations: HMAC, BBS+GGM



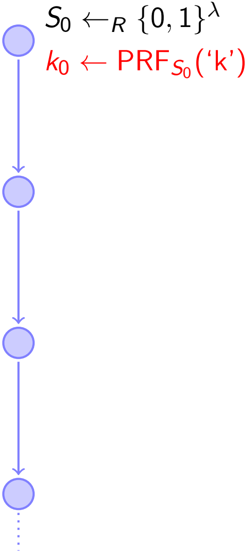
PRF-based KAS construction for chains

Remaining challenge: KAS for chains

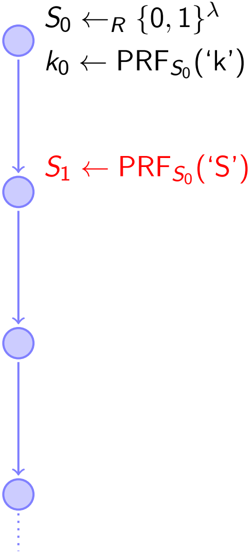
- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathrm{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and ‘k’, ‘S’ $\in D$
  - instantiations: HMAC, BBS+GGM



PRF-based KAS construction for chains



Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathrm{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and ‘k’, ‘S’ $\in D$
  - instantiations: HMAC, BBS+GGM

PRF-based KAS construction for chains

Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathrm{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and 'k', 'S' $\in D$
  - instantiations: HMAC, BBS+GGM

$S_0 \leftarrow_R \{0,1\}^\lambda$

$k_0 \leftarrow \mathrm{PRF}_{S_0}(\text{'k'})$

$S_1 \leftarrow \mathrm{PRF}_{S_0}(\text{'S'})$

$k_1 \leftarrow \mathrm{PRF}_{S_1}(\text{'k'})$

PRF-based KAS construction for chains

Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathrm{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and 'k', 'S' $\in D$
  - instantiations: HMAC, BBS+GGM

$S_0 \leftarrow_R \{0,1\}^\lambda$

$k_0 \leftarrow \mathrm{PRF}_{S_0}(\text{'k'})$

$S_1 \leftarrow \mathrm{PRF}_{S_0}(\text{'S'})$

$k_1 \leftarrow \mathrm{PRF}_{S_1}(\text{'k'})$

$S_2 \leftarrow \mathrm{PRF}_{S_1}(\text{'S'})$

$k_2 \leftarrow \mathrm{PRF}_{S_2}(\text{'k'})$

$S_3 \leftarrow \mathrm{PRF}_{S_2}(\text{'S'})$

$k_3 \leftarrow \mathrm{PRF}_{S_3}(\text{'k'})$

PRF-based KAS construction for chains

Remaining challenge: KAS for chains

- [AtaBlaFazFri05]
  - PRF-based, possibly insecure
- [CraDauMar10]
  - factoring-based construction (w/o proof)
- [FrePat11]
  - factoring-based construction (w/ proof)

Construction based on pseudorandom functions

- for totally ordered access graphs
- $\mathsf{PRF}_k : D \to R$
  - $K = R = \{0,1\}^\lambda$ and 'k', 'S' $\in D$
  - instantiations: HMAC, BBS+GGM

Theorem

Our PRF-based scheme offers S-KI-ST security, assuming security of PRF.

$$S_0 \leftarrow_R \{0,1\}^\lambda$$
$$k_0 \leftarrow \mathsf{PRF}_{S_0}(\text{'k'})$$

$$S_1 \leftarrow \mathsf{PRF}_{S_0}(\text{'S'})$$
$$k_1 \leftarrow \mathsf{PRF}_{S_1}(\text{'k'})$$

$$S_2 \leftarrow \mathsf{PRF}_{S_1}(\text{'S'})$$
$$k_2 \leftarrow \mathsf{PRF}_{S_2}(\text{'k'})$$

$$S_3 \leftarrow \mathsf{PRF}_{S_2}(\text{'S'})$$
$$k_3 \leftarrow \mathsf{PRF}_{S_3}(\text{'k'})$$

FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathrm{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathrm{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security

FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathsf{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathsf{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security

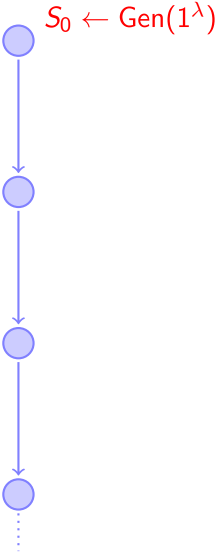
FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathsf{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathsf{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security



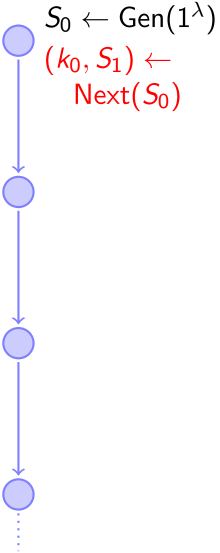
FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathrm{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathrm{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security



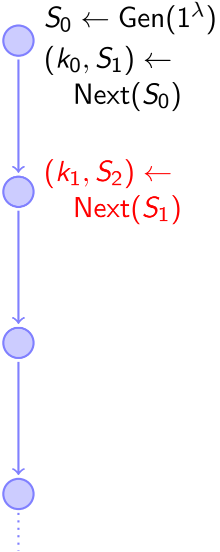
FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathsf{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathsf{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security



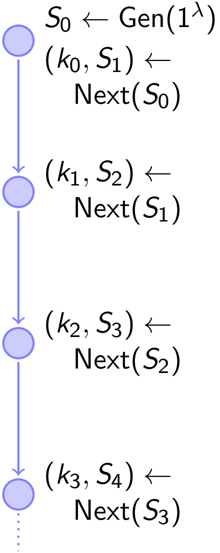
FSPRG-based KAS construction for chains

Construction based on FSPRGs

- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, . . .
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathsf{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathsf{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security



FSPRG-based KAS construction for chains

Construction based on FSPRGs

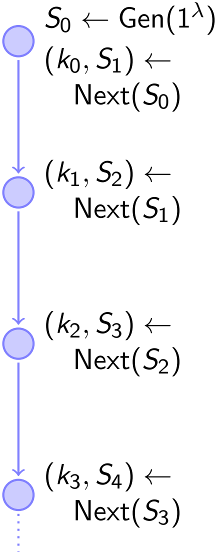
- forward-secure stateful PRGs [BelYee03]
- constructions based on HMAC, AES, BBS, ...
- generalizes [FrePat11]

Functionality of FSPRGs

- $\mathsf{Gen}(1^\lambda)$ (randomized)
  - output: initial state St_0
- $\mathsf{Next}(St_i)$ (deterministic)
  - output: string $Out_i \in \{0,1\}^\lambda$, state St_{i+1}
- $(Out_0, Out_1, \ldots)$ pseudorandom sequence
- forward security

Theorem

Our FSPRG-based scheme offers S-KI-ST security, assuming security of FSPRG.



Conclusion

Our contributions

- new security model
  - we introduced new security definition, fixing problems of previous ones
- security analysis of chain-based construction
  - we proved generic security of the chain partition method
- new constructions of KAS
  - we constructed highly efficient KAS, based on PRFs, (FS)PRGs, ...
  - we formally established their security

Conclusion

Our contributions

- new security model
  - we introduced new security definition, fixing problems of previous ones
- security analysis of chain-based construction
  - we proved generic security of the chain partition method
- new constructions of KAS
  - we constructed highly efficient KAS, based on PRFs, (FS)PRGs, ...
  - we formally established their security

Future work

- are established schemes secure in our stronger model?
- 'tree partition' vs. chain partition
- investigate reduction of [D'ArSanFerMas10]

Bibliography

- [AklTay83] *Cryptographic solution to a problem of access control in a hierarchy*, Akl, Taylor, ACM Trans. Comput. Syst.
- [AtaBlaFazFri05] *Dynamic and efficient key management for access hierarchies*, Atallah, Blanton, Fazio, Frikken, CCS & ACM Trans. Inf. Syst. Secur.
- [D'ArSanFerMas10] *Variations on a theme by Akl and Taylor: Security and tradeoffs*, D'Arco, De Santis, Ferrara, Masucci, Theor. Comput. Sci.
- [CraMarWil06] *On Key Assignment for Hierarchical Access Control*, Crampton, Martin, Wild, CSFW
- [CraDauMar10] *Constructing Key Assignment Schemes from Chain Partitions*, Crampton, Daud, Martin, DBSec
- [FrePat11] *Provably Secure Key Assignment Schemes from Factoring*, Freire, Paterson, ACISP
- [BelYee03] *Forward-Security in Private-Key Cryptography*, Bellare, Yee, CT-RSA

Randomized Partial Checking Revisited

Shahram Khazaei*
Douglas Wikström**

*Sharif University, Teheran, Iran (work done at KTH)
**KTH Royal Institute of Technology, Stockholm, Sweden

February 27, 2013

Voting systems vs mix-nets

1. Servers/trustees run **distributed key generation** protocol to generate a public key *pk* for which the secret key is verifiably secret shared.

Voting systems vs mix-nets

1. Servers/trustees run **distributed key generation** protocol to generate a public key *pk* for which the secret key is verifiably secret shared.

2. Voters form encryptions of their votes using the public key *pk* and a verifiable **submission scheme**.

Voting systems vs mix-nets

1. Servers/trustees run **distributed key generation** protocol to generate a public key *pk* for which the secret key is verifiably secret shared.

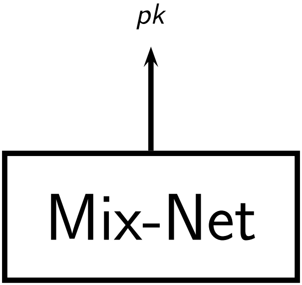
2. Voters form encryptions of their votes using the public key *pk* and a verifiable **submission scheme**.

3. Servers/trustees execute a **mix-net** to simultaneously permute and decrypt the ciphertexts.

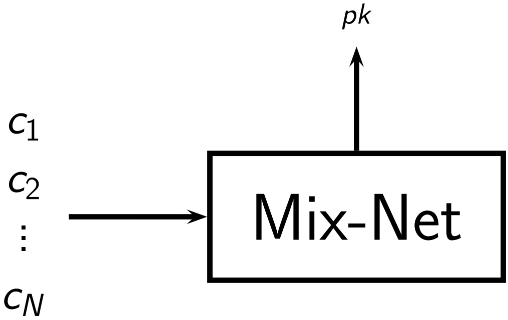
Mix-net [Chaum]

$\boxed{\text{Mix-Net}}$

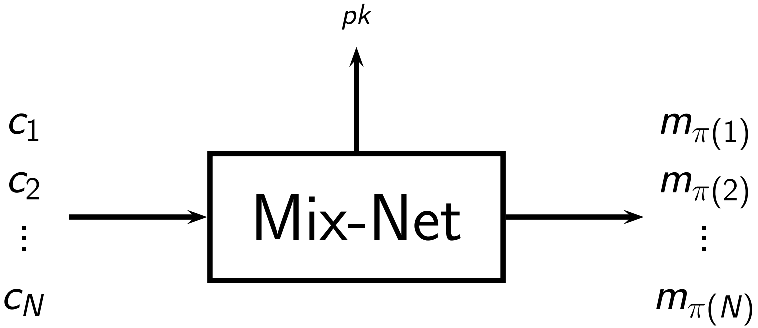
Mix-net [Chaum]



Mix-net [Chaum]



Mix-net [Chaum]



Simultanously decrypt and randomly permute ciphertexts.

Re-encryption Mix-Nets

Homomorphic cryptosystem

$$\mathsf{E}_{pk}(m_0) \times \mathsf{E}_{pk}(m_1) = \mathsf{E}_{pk}(m_0 m_1)$$

Homomorphic cryptosystem

$$\mathsf{E}_{pk}(m_0, r_0) \times \mathsf{E}_{pk}(m_1, r_1) = \mathsf{E}_{pk}(m_0 m_1, r_0 + r_1)$$

Homomorphic cryptosystem

$$\mathsf{E}_{pk}(m_0, r_0) \times \mathsf{E}_{pk}(m_1, r_1) = \mathsf{E}_{pk}(m_0 m_1, r_0 + r_1)$$

This property can be used to **re-encrypt** ciphertexts

$$\mathsf{E}_{pk}(m_0, r_0) \times \mathsf{E}_{pk}(1, r_1) = \mathsf{E}_{pk}(m_0, r_0 + r_1)$$

Re-encryption mix-net

Mix-servers $M_1, \ldots, M_k$.

Execution of mix-net:

1. Run distributed key generation protocol to generate joint El Gamal public key pk.
2. Simultaneously re-encrypt and permute all ciphertexts.
3. Run distributed decryption protocol on the re-encrypted and permuted ciphertexts.

Re-encryption mix-net

$$\boxed{M_1} \qquad \boxed{M_2} \qquad \cdots \qquad \boxed{M_k}$$

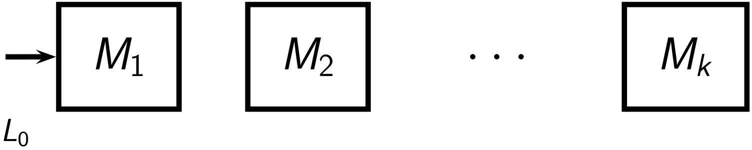
Re-encryption mix-net

L_0 = list of input ciphertexts

$$\boxed{M_1} \quad \boxed{M_2} \quad \cdots \quad \boxed{M_k}$$

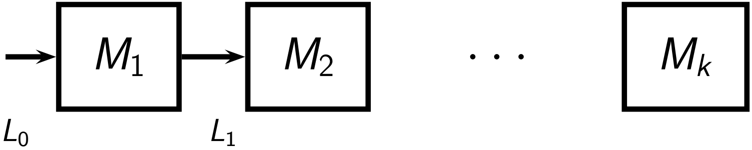
Re-encryption mix-net

L_0 = list of input ciphertexts



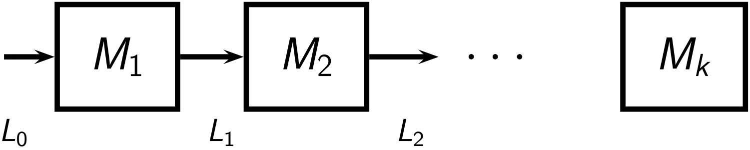
Re-encryption mix-net

L_0 = list of input ciphertexts



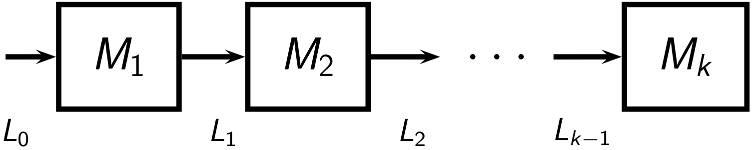
Re-encryption mix-net

L_0 = list of input ciphertexts



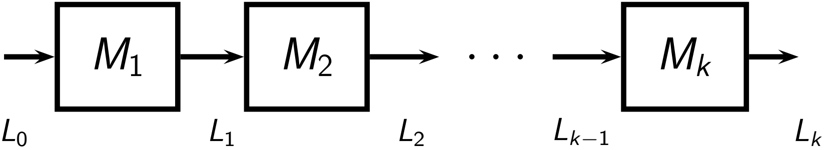
Re-encryption mix-net

L_0 = list of input ciphertexts



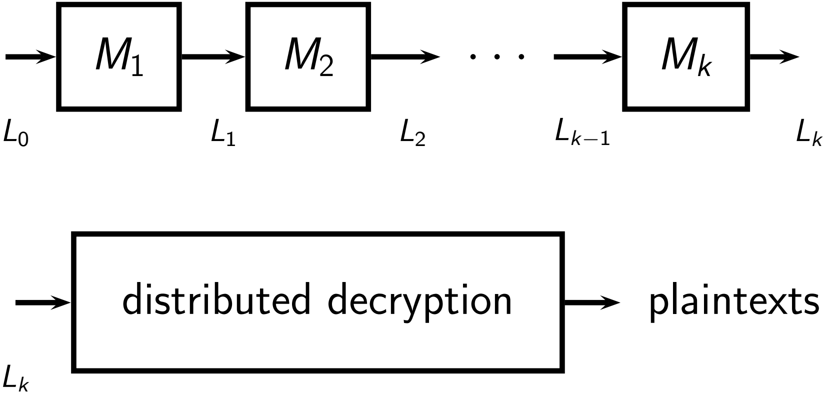
Re-encryption mix-net

L_0 = list of input ciphertexts



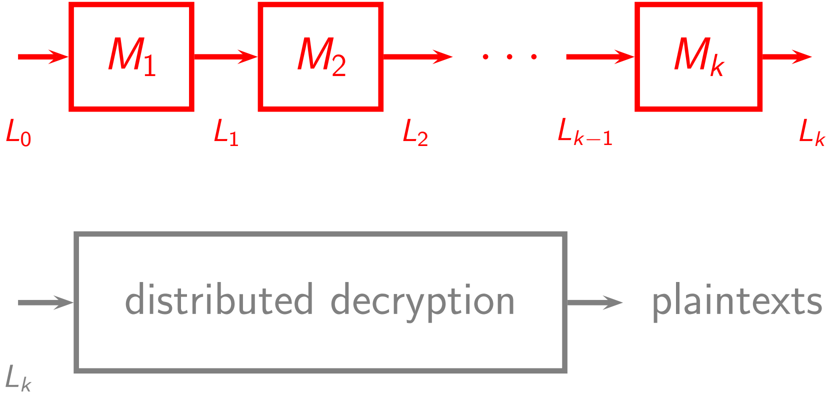
Re-encryption mix-net

L_0 = list of input ciphertexts



Re-encryption mix-net

L_0 = list of input ciphertexts



Re-encryption mix-net

$$L_{j-1} = (c_{j-1,1}, \ldots, c_{j-1,N})$$

$$L_{j-1} \longrightarrow \boxed{M_j} \longrightarrow L_j$$

Re-encryption mix-net

$$L_{j-1} = (c_{j-1,1}, \ldots, c_{j-1,N})$$

$$L_{j-1} \longrightarrow \boxed{M_j} \longrightarrow L_j$$

$c_{j-1,i}$ (input ciphertext)

Re-encryption mix-net

$$L_{j-1} = (c_{j-1,1}, \ldots, c_{j-1,N})$$

$$L_{j-1} \longrightarrow \boxed{M_j} \longrightarrow L_j$$

$$\mathrm{E}_{pk}(1, r_{j,i}) \times c_{j-1,i} \quad \text{(re-encrypt)}$$

Re-encryption mix-net

$$L_{j-1} = (c_{j-1,1}, \ldots, c_{j-1,N})$$

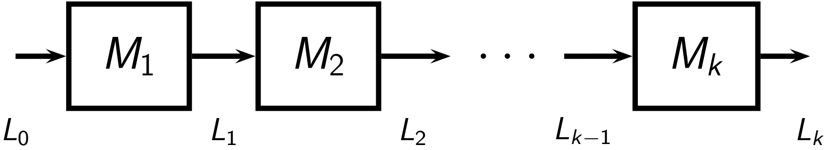
$$L_{j-1} \longrightarrow \boxed{M_j} \longrightarrow L_j$$

$$c_{j,i} = \mathsf{E}_{pk}(1, r_{j,\pi_j(i)}) \times c_{j-1,\pi_j(i)}$$ (re-encrypt and permute)

$$L_j = (c_{j,1}, \ldots, c_{j,N})$$

Re-encryption mix-net

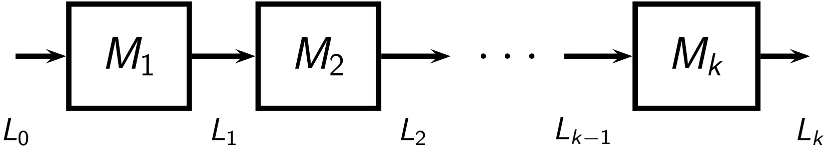
L_0 = list of input ciphertexts



What if a mix-server is malicious?

Re-encryption mix-net

L_0 = list of input ciphertexts



Each mix-server proves that it behaves!

Randomized Partial Checking

Randomized partial checking: basic idea

Proposed by Jakobsson, Juels and Rivest USENIX 2002.

Each mix-server is challenged to **reveal a little** about what it did. **Privacy** should still be **preserved jointly** by the mix-servers.

Motivation

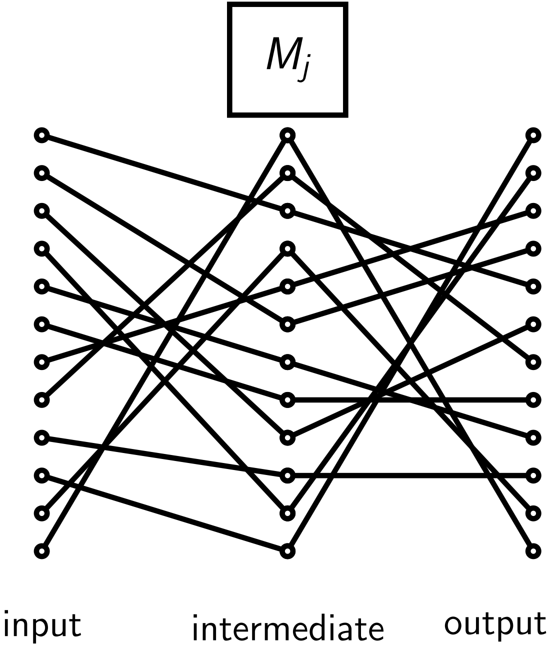
- Beautiful idea with potential to solve the problem for **any cryptosystem**.
- Used in numerous papers about **verifiable** electronic voting systems.
- Implemented in **Civitas** by Clarkson, ... at Cornell.
- Similar idea used in **Scantegrity** developed by Chaum, Rivest, ... at MIT, Maryland, Ottowa, Waterloo, George Washington. **Used** in real municipal elections in Takoma Park.
- Implemented i several other research projects with non-released code (we were generously given access).

Motivation

- Beautiful idea with potential to solve the problem for **any cryptosystem**.
- Used in numerous papers about **verifiable** electronic voting systems.
- Implemented in **Civitas** by Clarkson, ... at Cornell.
- Similar idea used in **Scantegrity** developed by Chaum, Rivest, ... at MIT, Maryland, Ottowa, Waterloo, George Washington. **Used** in real municipal elections in Takoma Park.
- Implemented i several other research projects with non-released code (we were generously given access).

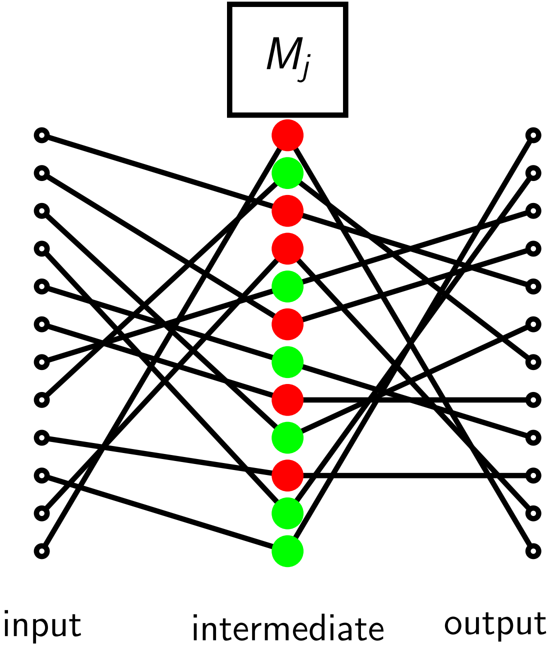
I had a GOOD gut feeling about this!

Randomized partial checking



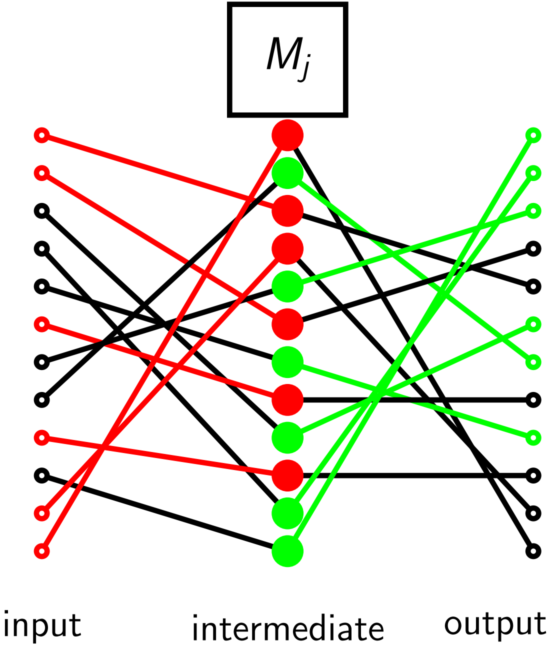
Randomized partial checking

1. Partition



Randomized partial checking

1. Partition
2. Open



More details

Before challenge, M_j also commits to:

- Origin index of each intermediate ciphertext.
- Destination index of each intermediate ciphertext.

More details

Before challenge, M_j also commits to:

- Origin index of each intermediate ciphertext.
- Destination index of each intermediate ciphertext.

Open chosen commitments along with randomness.

More details

Before challenge, M_j also commits to:

- Origin index of each intermediate ciphertext.
- Destination index of each intermediate ciphertext.

Open chosen commitments along with randomness.

Attacks works as if this was not in place.

Attacks

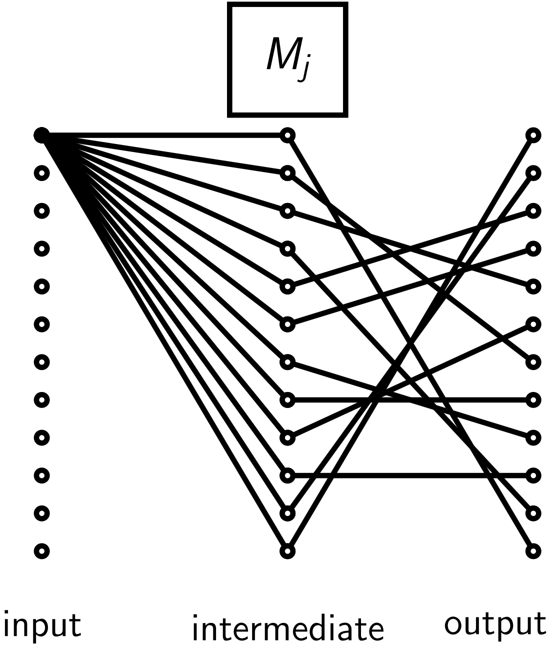
Attack on soundness

It is never verified that the revealed indices are distinct.

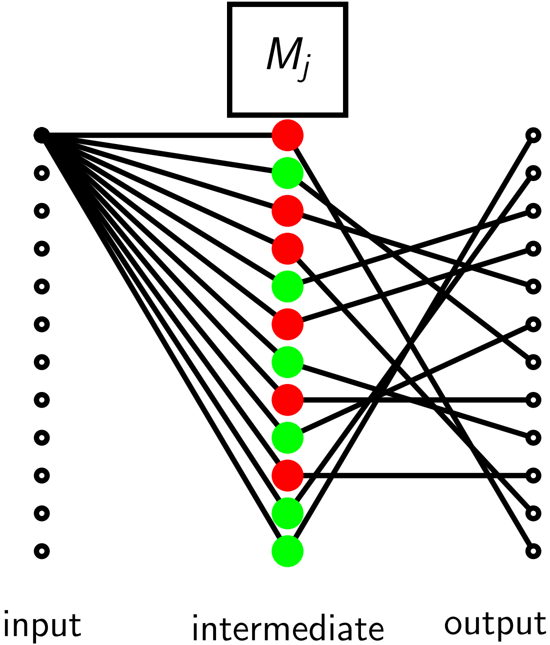
Thus, nothing prevents M_j to replace all origin indices by one.

This allows **replacing all ciphertexts without detection.**

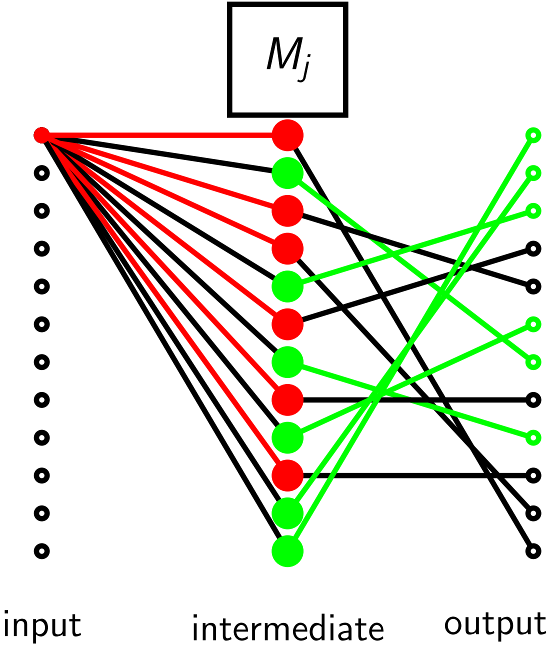
Attack on soundness



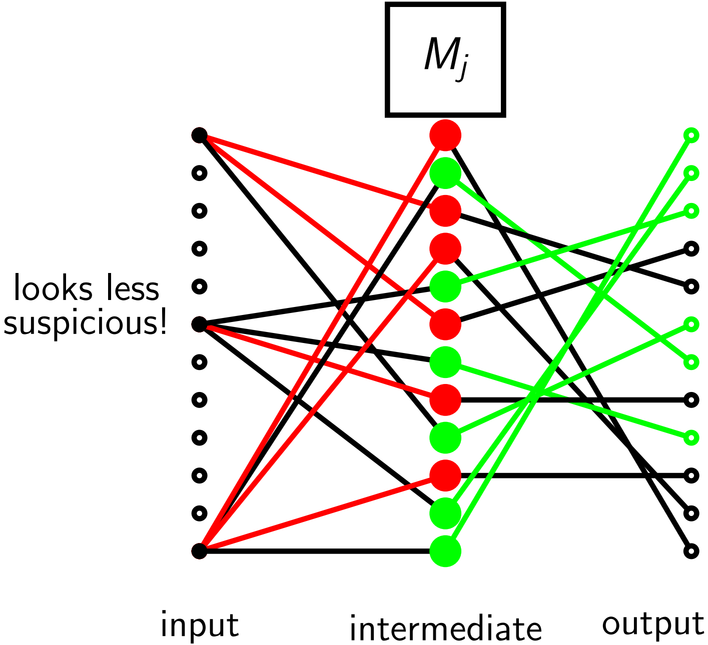
Attack on soundness



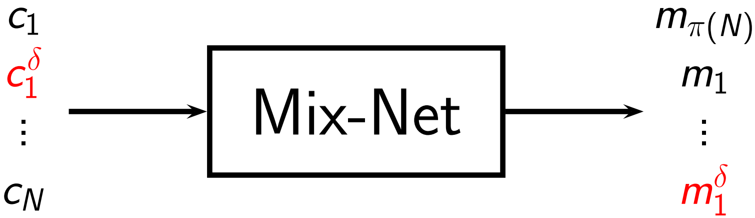
Attack on soundness



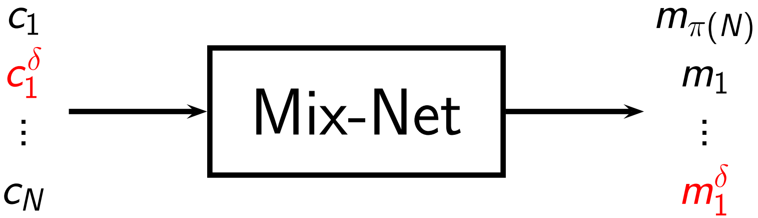
Attack on soundness



Recall Pfitzmann's attack

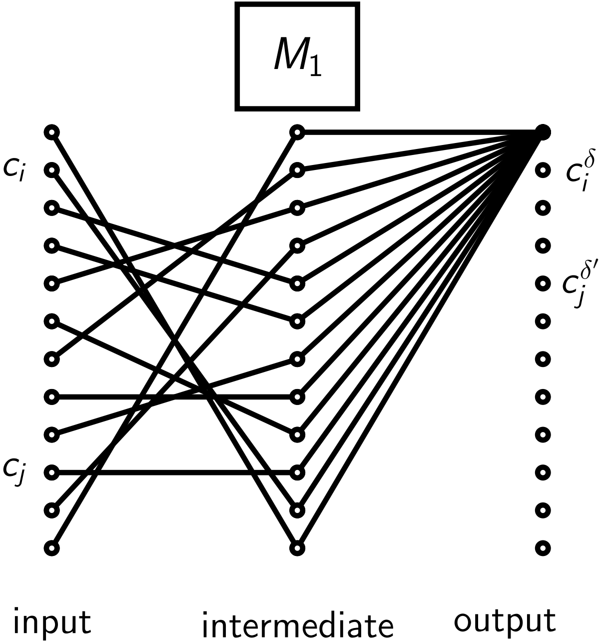


Recall Pfitzmann's attack



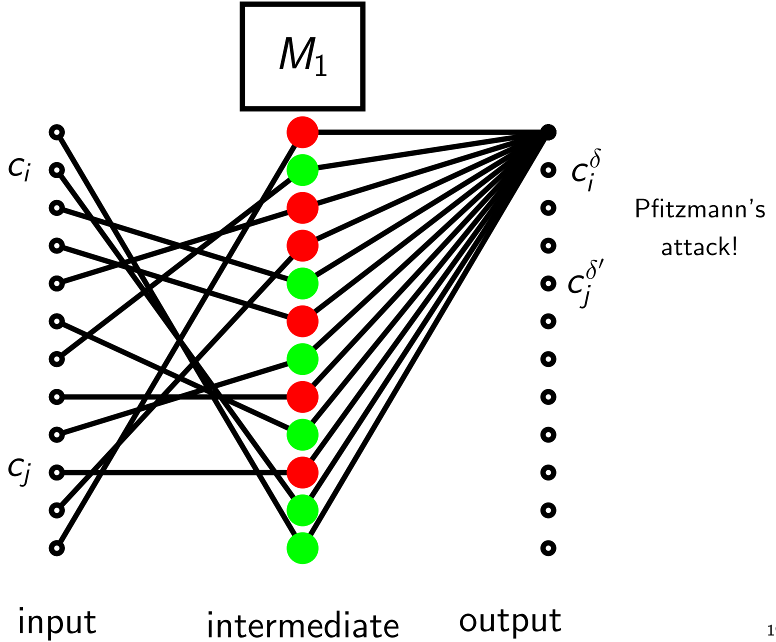
Force non-malleability, e.g., ZKPoK!

Attack on privacy

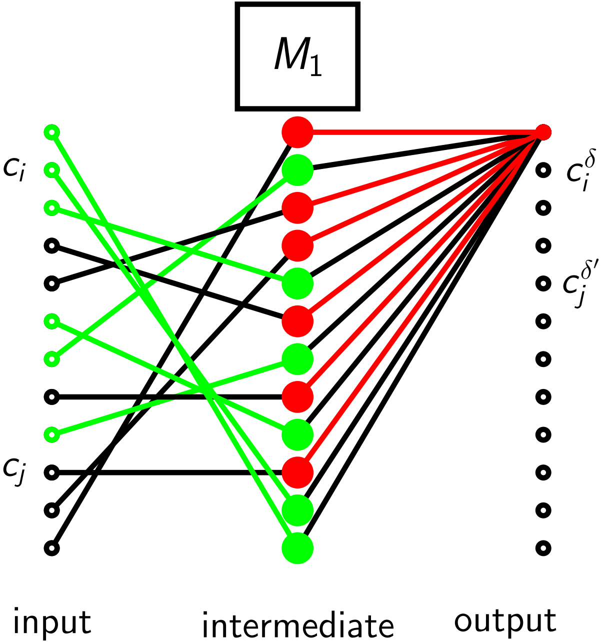


Pfitzmann's attack!

Attack on privacy

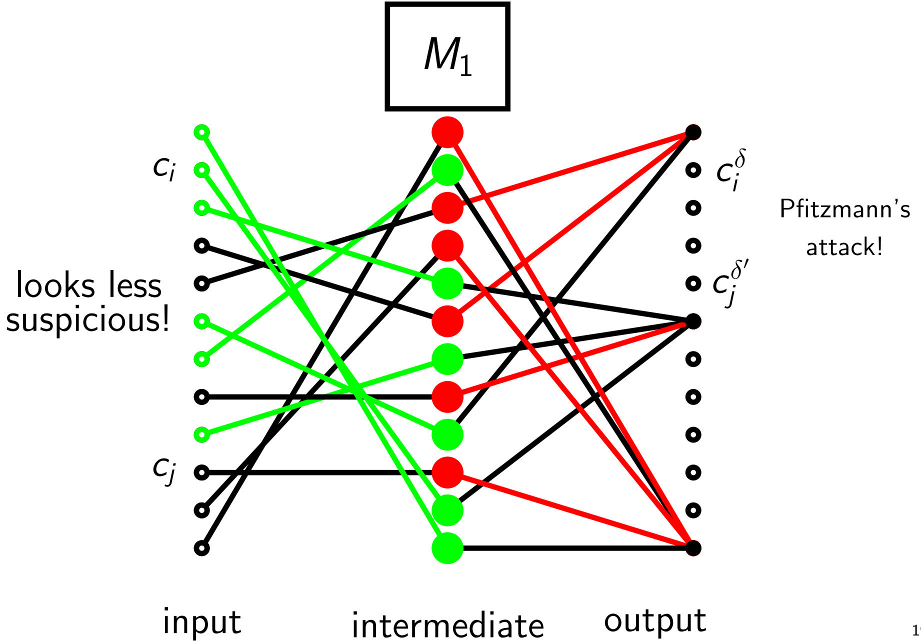


Attack on privacy



Pfitzmann's attack!

Attack on privacy



Privacy

Combined with Pfitzmann we can break privacy of almost everybody, but it would be noticed in the output (one-off attack).

...some additional observations.

Yeah, yeah, a very serious bug, but can it be patched?

Yes, it can be patched in the obvious way, but...

Yeah, yeah, a very serious bug, but can it be patched?

Yes, it can be patched in the obvious way, but...

Claim. c ciphertexts can be replaced with probability roughly $(1/2)^c$.

Correct. c ciphertexts can be replaced with probability roughly $(3/4)^c$.

Yeah, yeah, a very serious bug, but can it be patched?

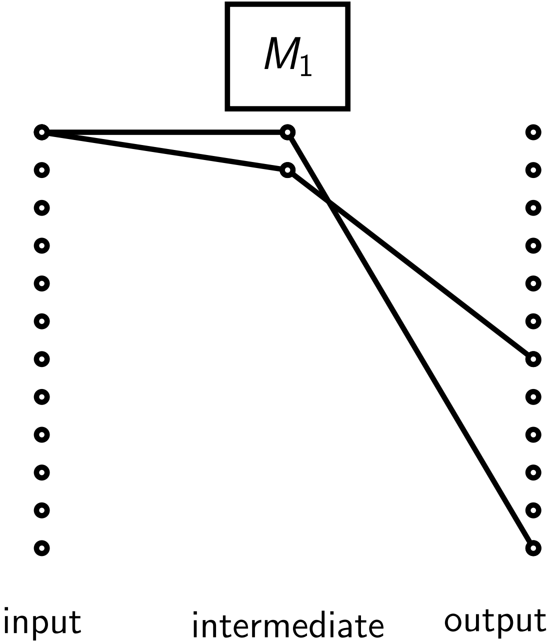
Yes, it can be patched in the obvious way, but...

Claim. c ciphertexts can be replaced with probability roughly $(1/2)^c$.

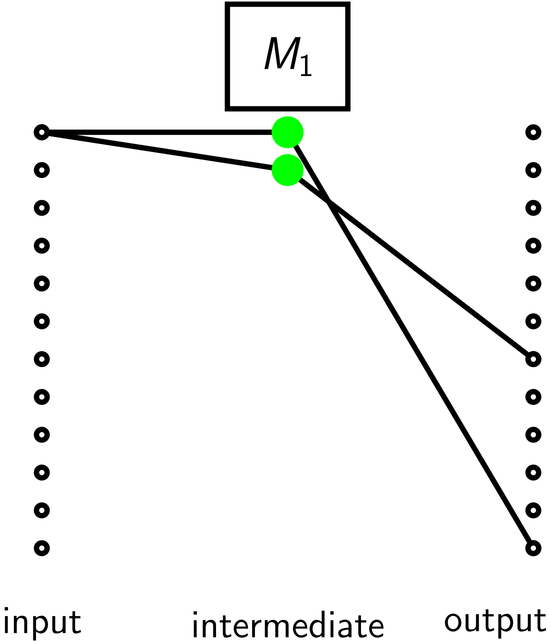
Correct. c ciphertexts can be replaced with probability roughly $(3/4)^c$.

Important! Universal verifiability can be violated for changes of c ciphertexts with work $O\big((4/3)^c\big)$ instead of $O(2^c)$.

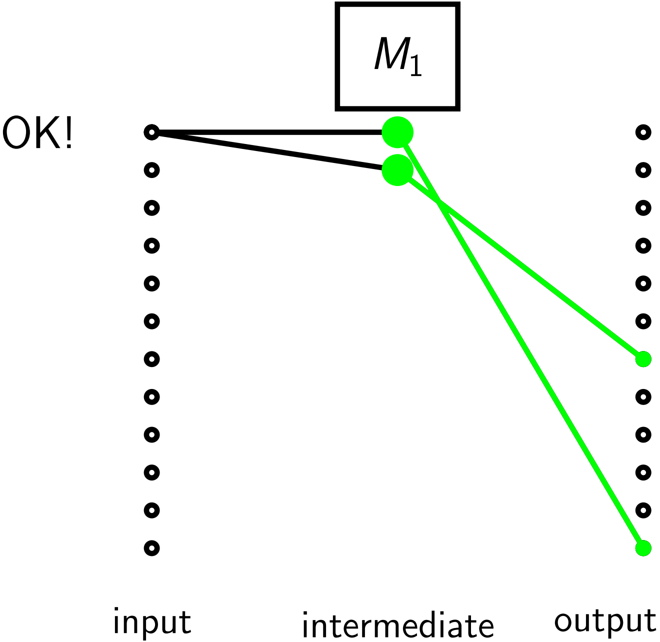
Wrong bound!



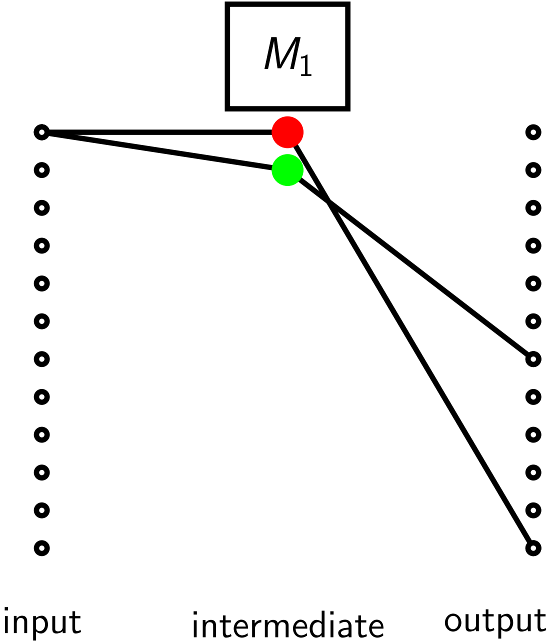
Wrong bound!



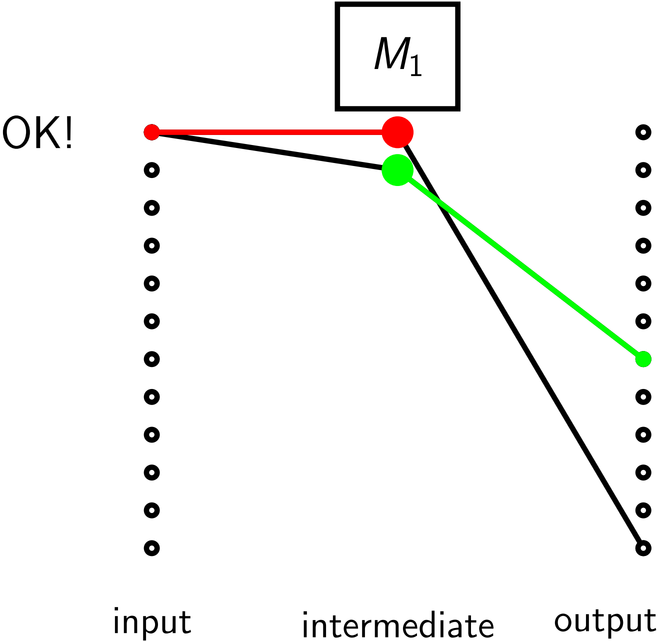
Wrong bound!



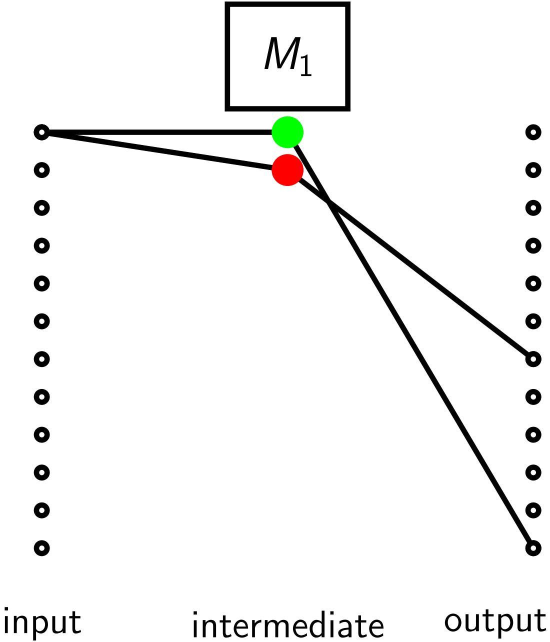
Wrong bound!



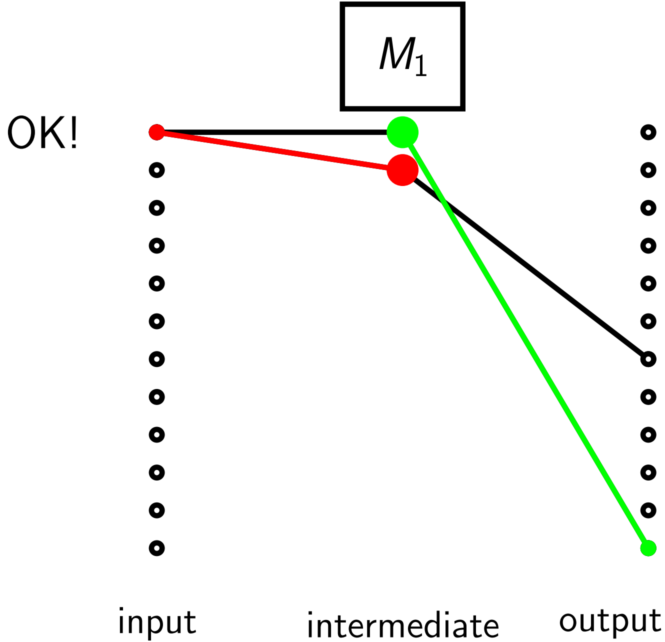
Wrong bound!



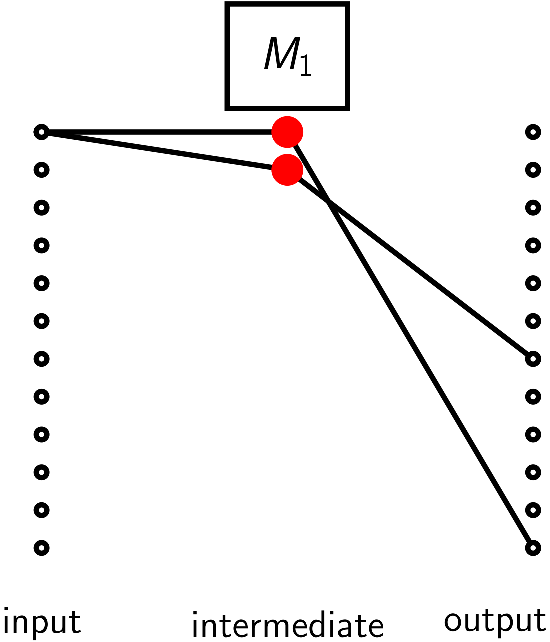
Wrong bound!



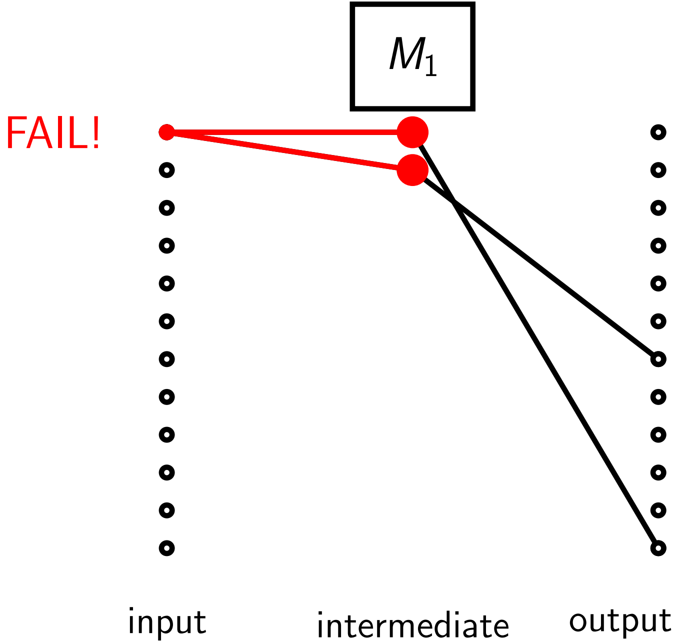
Wrong bound!



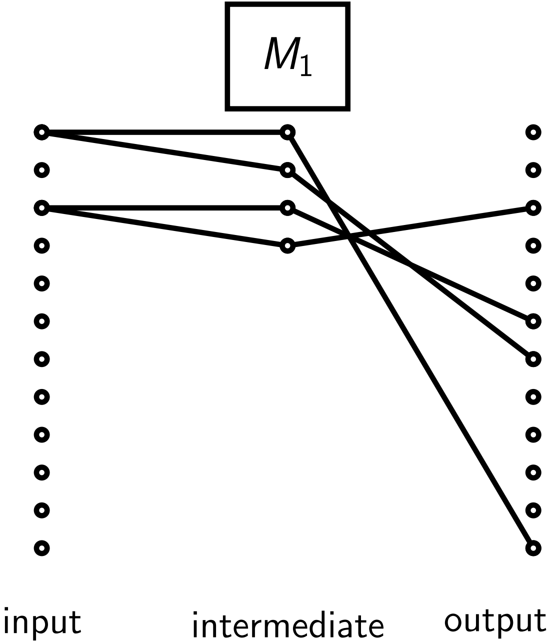
Wrong bound!



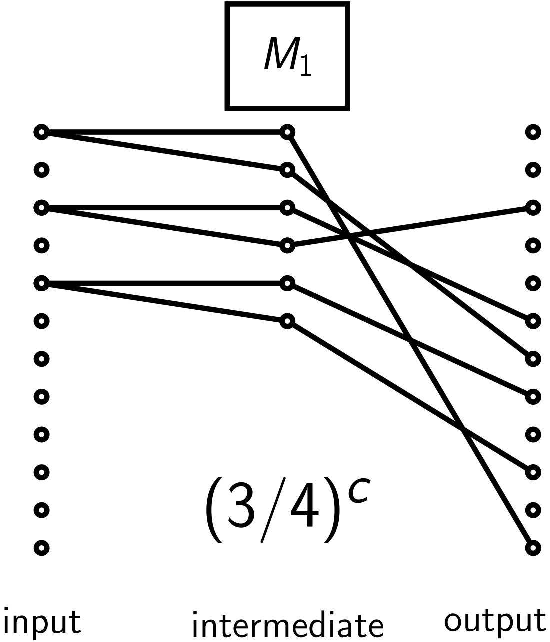
Wrong bound!



Wrong bound!



Wrong bound!



Conclusion

Current work

We are writing down a rigorous proof of security for mix-nets with randomized partial checking.

Non-standard weaker security properties than claimed and hairy proof, but it works almost as expected.

Recent attacks

Flaws have been found in all “big” verifiable systems!

- Scytl mix-net (non-fatal vulnerabilities) [KTW EVT/WOTE ’12]
- Helios (universal verifiability fails) [BPW Asiacrypt ’12]
- Civitas, Scantegrity, N.N.,... (soundness fails) [KW today]

Recent attacks

Flaws have been found in all “big” verifiable systems!

- Scytl mix-net (non-fatal vulnerabilities) [KTW EVT/WOTE ’12]
- Helios (universal verifiability fails) [BPW Asiacrypt ’12]
- Civitas, Scantegrity, N.N.,... (soundness fails) [KW today]

I am still optimistic about the whole thing!

What we need

Unless at least the counting in an electronic election scheme is provably sound/verifiable it should not be used.

We need **several independent** cryptographers and/or machines to verify the proofs of security.

We must spend more time scrutinizing the proposals of the community and the underlying assumptions.

Questions?