CS 578 – Cryptography

Prof. Michael Backes

MACs, Collision-Resistant Hash Functions, Combining Privacy and Integrity

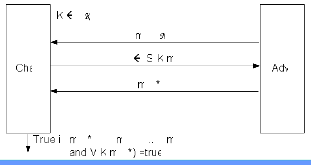
May 16, 2006

Administrative Announcements

- My office hours:
  - Monday 12:00-13:00
    (not 13:00-14:00, sorry if this caused confusions.)
  - Until the mid-term exam has been written:
    Additional office hour: Wednesday 11:00-12:00
- Handouts today:
  - Lecture notes, next exercise sheet
- Try to make the lecture notes available earlier:
  - Additional lecture notes today for Friday lecture

Recall: Definition of MAC

- Definition (MAC): A message authentication code (MAC) defined over $(\mathcal{K},\mathcal{M},\mathcal{T})$ is a pair (S,V) of efficient algorithms (S,V) where
  S: $\mathcal{K}\times \mathcal{M} \rightarrow \mathcal{T}$ and V: $\mathcal{K}\times \mathcal{M}\times \mathcal{T} \rightarrow \mathcal{B}ool$
  s.t. for all m $\in \mathcal{M}$, K $\in \mathcal{K}$: (t ← S(K,m)) ➔ (V(K,m,t) = true)
- Definition (Secure MACs, intuitively):

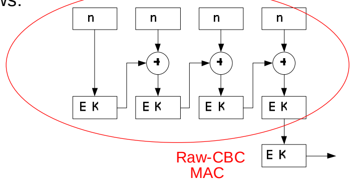


Recall: PRFs and MACs

- Any PRF with sufficiently large range is a secure MAC
- Given a small PRF (MAC), we compute a big PRF (MAC):
- Last Lecture:
  - CBC-MAC, used by banks, etc., sequential
  - PMAC = Parallel MAC, not used in practice, incremental
- Today: HMAC, used in lots of Internet protocols, incremental, build on collision-resistant hash functions (CRHFs)

Recall: (Encrypted) CBC-MAC

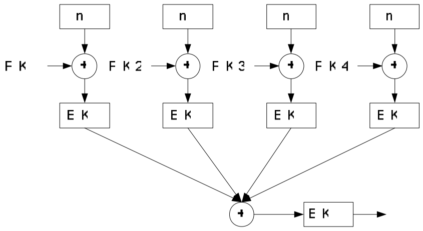
- Let E be a PRF over $(\mathcal{K},\mathcal{X},\mathcal{X})$, e.g., AES
- Define a PRF E^{CBC} (and thus also a MAC) as follows:



- E^{CBC} is a function from $\mathcal{K}^2 \times \mathcal{X}^L \rightarrow \mathcal{X}$

Recall: PMAC – Parallel MAC

- Usual problem with CBC: sequential
- (One) remedy: PMAC – Parallel MAC



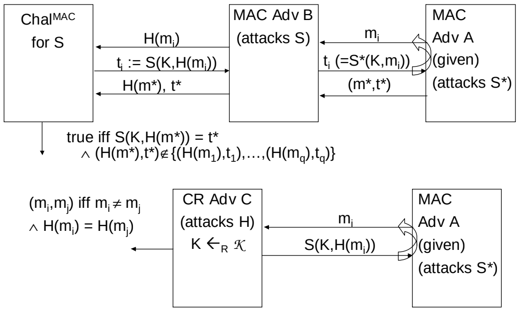
Recall: Hash Functions

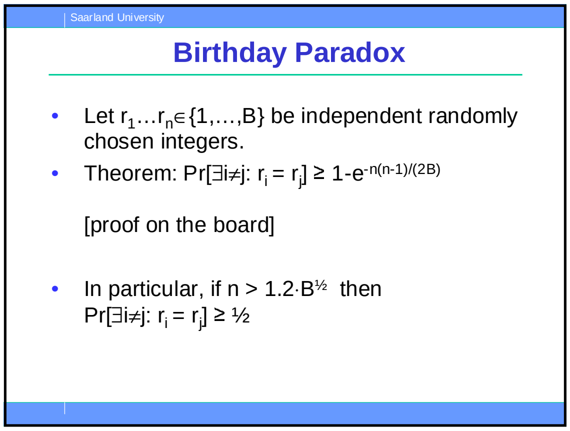
- Let H : M → T be a hash function (non-keyed) (often H : {0,1}* → $\{0,1\}^n$)
- A collision for H is a tuple (m_1,m_2) with $H(m_1)=H(m_2) \wedge m_1 \neq m_2$.
- “Definition” (Collision Resistant Hash Function, CRHF): A hash function is collision resistant if no algorithm is known that finds a collision for H in suitable time.
- Remark: Defining that “no efficient adversary exists that finds a collision” cannot be fulfilled

CRHFs and MACs

- Construction of big-MACs from small-MACs and CRHFs:
  - Let I=(S,V) be a secure MAC for short messages (e.g., AES) over $(\mathcal{K},\mathcal{M},\mathcal{T})$
  - Let H be collision-resistance hash function: H: $\mathcal{M}^{big} \rightarrow \mathcal{M}$
- Definition (big-MACs from small-MACs): Let I*=(S*,V*) be a MAC over $(\mathcal{K},\mathcal{M}^{big},\mathcal{T})$ where
  - S*(K,m) := S(K,H(m))
  - V*(K,m,t) = true iff V(K,H(m),t) = true
- Theorem: I* is a secure MAC.
- → AES(K,H(m)) is a secure MAC if H is a CRHF.
- How to build collision-resistant hash functions?

Proof Sketch



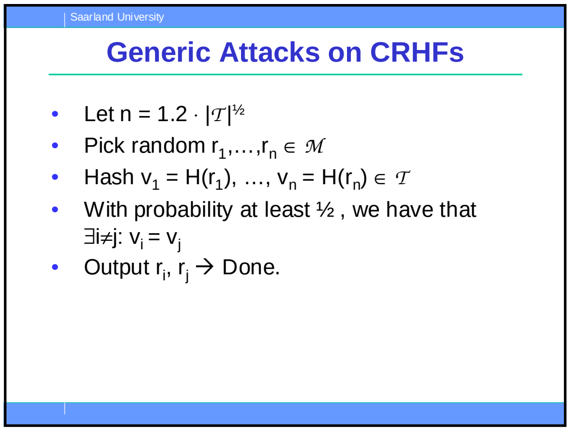


Birthday Paradox

- Let $r_1 \ldots r_n \in \{1,\ldots,B\}$ be independent randomly chosen integers.
- Theorem: $\Pr[\exists i \neq j: r_i = r_j] \geq 1 - e^{-n(n-1)/(2B)}$

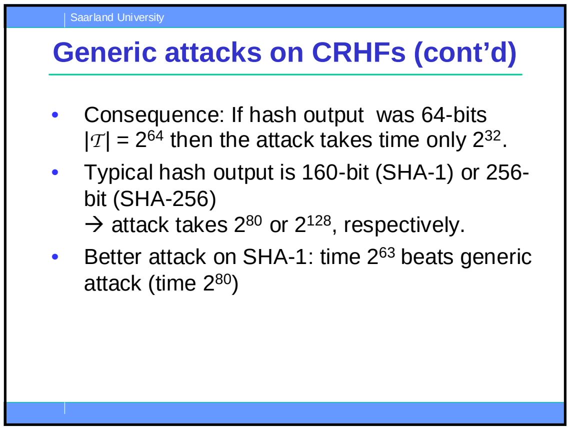
  [proof on the board]

- In particular, if $n > 1.2 \cdot B^{½}$ then $\Pr[\exists i \neq j: r_i = r_j] \geq ½$



Generic Attacks on CRHFs

- Let $n = 1.2 \cdot |\mathcal{T}|^{½}$
- Pick random $r_1,\ldots,r_n \in \mathcal{M}$
- Hash $v_1 = H(r_1), \ldots, v_n = H(r_n) \in \mathcal{T}$
- With probability at least ½ , we have that $\exists i \neq j: v_i = v_j$
- Output r_i, r_j → Done.



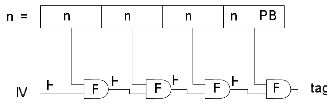
Generic attacks on CRHFs (cont'd)

- Consequence: If hash output was 64-bits $|\mathcal{T}| = 2^{64}$ then the attack takes time only 2^{32}.
- Typical hash output is 160-bit (SHA-1) or 256-bit (SHA-256)

  → attack takes 2^{80} or 2^{128}, respectively.
- Better attack on SHA-1: time 2^{63} beats generic attack (time 2^{80})

Construcing CRHF

- Merkle-Damgard (iterated construction)



- F: $\{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$: compression function.
- H_i are called chaining variables
- IV is the initial value
- PB is padding block

Padding for Merkle-Damgard

- Padding Block PB:

$$\boxed{1000\ldots\ldots\ldots.0}\underbrace{\boxed{\text{msg\_len}}}_{\text{64 bits}}$$

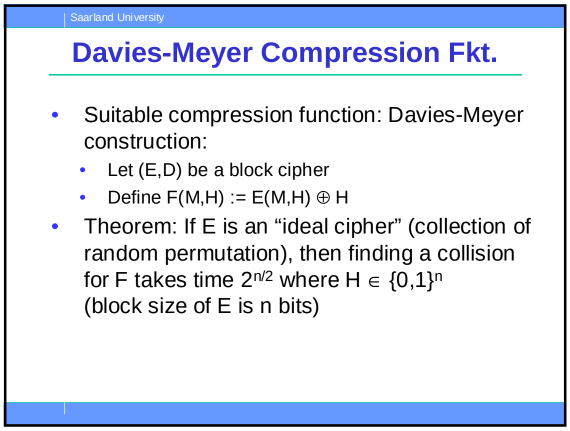
such that m[last] || PB is in $\{0,1\}^b$

Merkle-Damgard (cont'd)

- Lemma: If compression function F is collision resistant, then Merkle-Damgard hash (MD hash) is also collision-resistant.

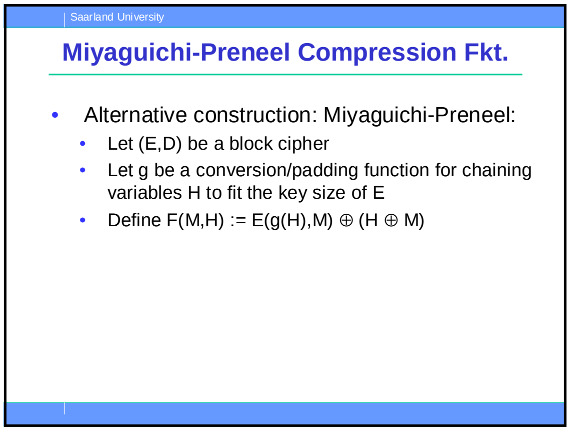
  [proof on the board]

→ To build collision-resistant hash functions, we only need small compression functions



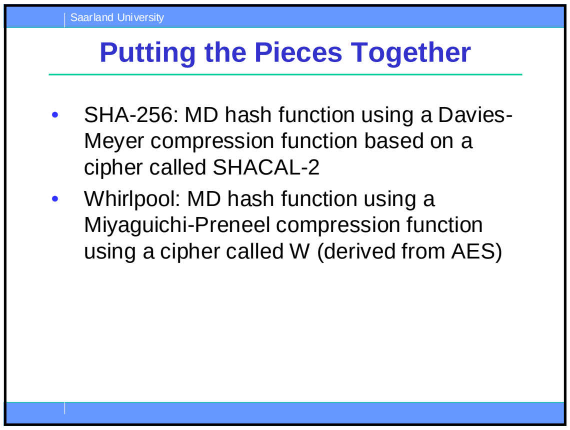
Davies-Meyer Compression Fkt.

- Suitable compression function: Davies-Meyer construction:
  - Let (E,D) be a block cipher
  - Define F(M,H) := E(M,H) ⊕ H
- Theorem: If E is an “ideal cipher” (collection of random permutation), then finding a collision for F takes time $2^{n/2}$ where $H \in \{0,1\}^n$ (block size of E is n bits)



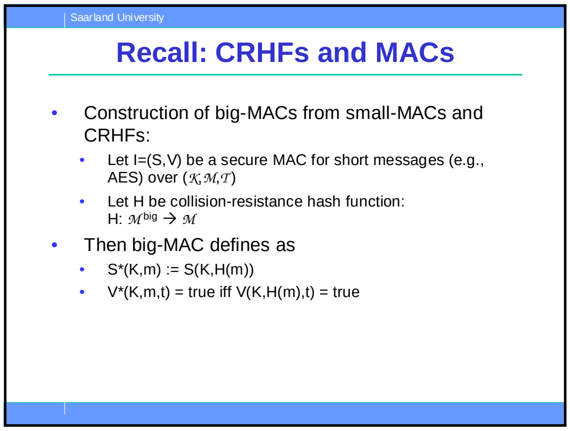
Miyaguichi-Preneel Compression Fkt.

- Alternative construction: Miyaguichi-Preneel:
  - Let (E,D) be a block cipher
  - Let g be a conversion/padding function for chaining variables H to fit the key size of E
  - Define F(M,H) := E(g(H),M) ⊕ (H ⊕ M)



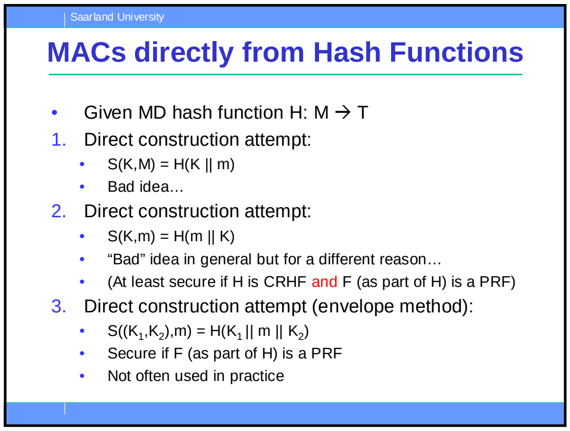
Putting the Pieces Together

- SHA-256: MD hash function using a Davies-Meyer compression function based on a cipher called SHACAL-2
- Whirlpool: MD hash function using a Miyaguichi-Preneel compression function using a cipher called W (derived from AES)



Recall: CRHFs and MACs

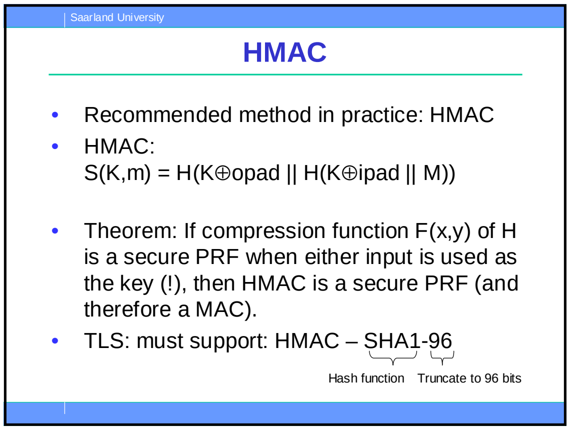
- Construction of big-MACs from small-MACs and CRHFs:
  - Let I=(S,V) be a secure MAC for short messages (e.g., AES) over ($\mathcal{K},\mathcal{M},\mathcal{T}$)
  - Let H be collision-resistance hash function: H: $\mathcal{M}^{big} \rightarrow \mathcal{M}$
- Then big-MAC defines as
  - S*(K,m) := S(K,H(m))
  - V*(K,m,t) = true iff V(K,H(m),t) = true



MACs directly from Hash Functions

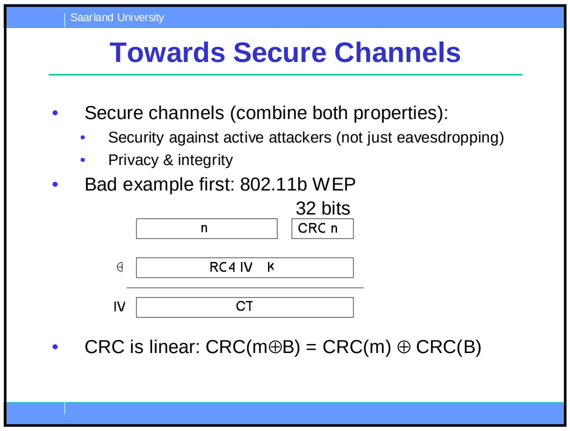
- Given MD hash function H: M → T

1. Direct construction attempt:
   - S(K,M) = H(K || m)
   - Bad idea…
2. Direct construction attempt:
   - S(K,m) = H(m || K)
   - "Bad" idea in general but for a different reason…
   - (At least secure if H is CRHF and F (as part of H) is a PRF)
3. Direct construction attempt (envelope method):
   - S((K_1,K_2),m) = H(K_1 || m || K_2)
   - Secure if F (as part of H) is a PRF
   - Not often used in practice



HMAC

- Recommended method in practice: HMAC
- HMAC:
  S(K,m) = H(K⊕opad || H(K⊕ipad || M))
- Theorem: If compression function F(x,y) of H is a secure PRF when either input is used as the key (!), then HMAC is a secure PRF (and therefore a MAC).
- TLS: must support: HMAC – $\underbrace{\text{SHA1}}_{\text{Hash function}}\text{-}\underbrace{\text{96}}_{\text{Truncate to 96 bits}}$

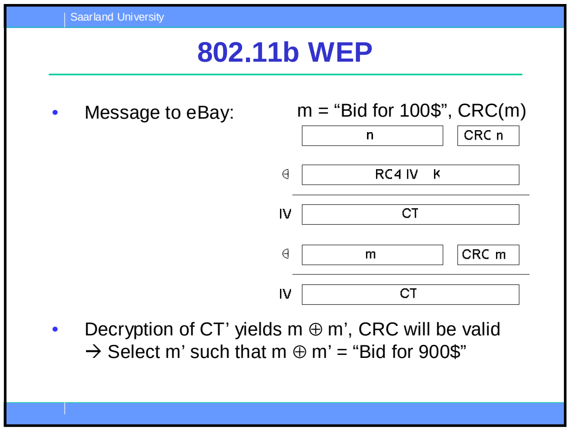


Towards Secure Channels

- Secure channels (combine both properties):
  - Security against active attackers (not just eavesdropping)
  - Privacy & integrity
- Bad example first: 802.11b WEP

32 bits
n
CRC n
⊕
RC4 IV K
IV
CT

- CRC is linear: CRC(m⊕B) = CRC(m) ⊕ CRC(B)

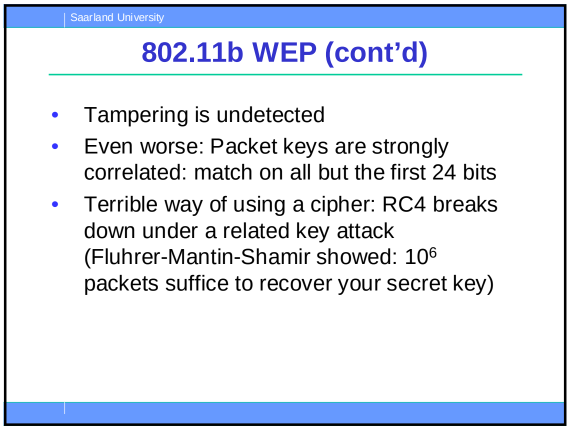


802.11b WEP

- Message to eBay: m = “Bid for 100$”, CRC(m)

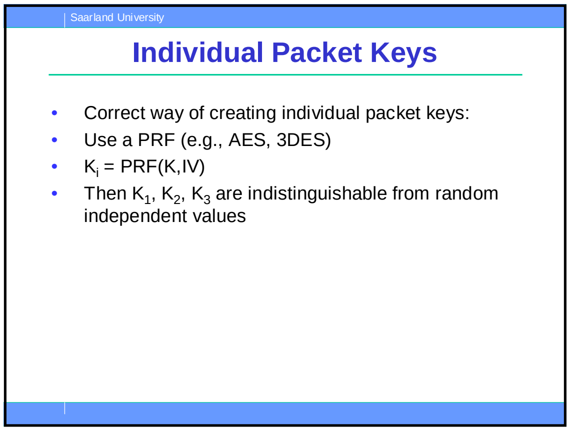
n
CRC n
⊕
RC4 IV K
IV
CT
⊕
m
CRC m
IV
CT

- Decryption of CT’ yields m ⊕ m’, CRC will be valid
  → Select m’ such that m ⊕ m’ = “Bid for 900$”



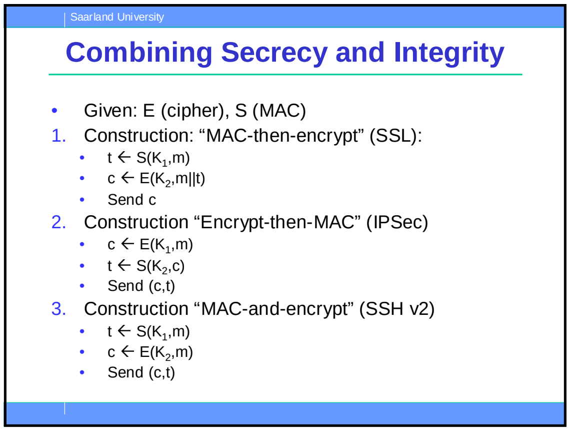
802.11b WEP (cont’d)

- Tampering is undetected
- Even worse: Packet keys are strongly correlated: match on all but the first 24 bits
- Terrible way of using a cipher: RC4 breaks down under a related key attack (Fluhrer-Mantin-Shamir showed: 10^6 packets suffice to recover your secret key)



Individual Packet Keys

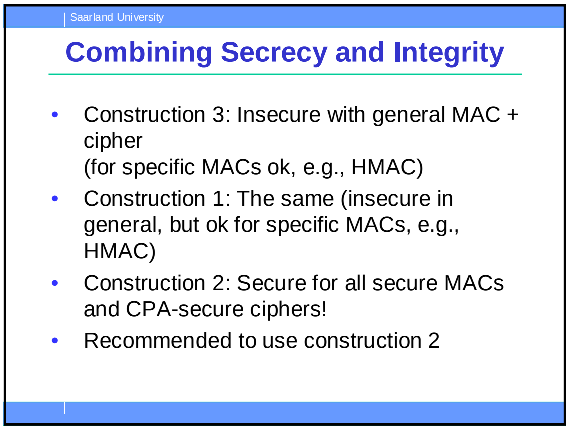
- Correct way of creating individual packet keys:
- Use a PRF (e.g., AES, 3DES)
- K_i = PRF(K,IV)
- Then K_1, K_2, K_3 are indistinguishable from random independent values



Combining Secrecy and Integrity

- Given: E (cipher), S (MAC)

1. Construction: “MAC-then-encrypt” (SSL):
   - t ← S(K_1,m)
   - c ← E(K_2,m||t)
   - Send c
2. Construction “Encrypt-then-MAC” (IPSec)
   - c ← E(K_1,m)
   - t ← S(K_2,c)
   - Send (c,t)
3. Construction “MAC-and-encrypt” (SSH v2)
   - t ← S(K_1,m)
   - c ← E(K_2,m)
   - Send (c,t)

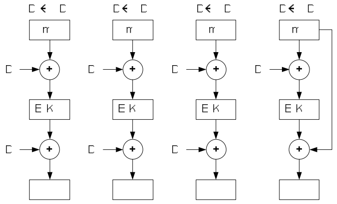


Combining Secrecy and Integrity

- Construction 3: Insecure with general MAC + cipher
  (for specific MACs ok, e.g., HMAC)
- Construction 1: The same (insecure in general, but ok for specific MACs, e.g., HMAC)
- Construction 2: Secure for all secure MACs and CPA-secure ciphers!
- Recommended to use construction 2

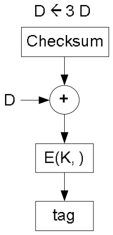
Offset Codebook (OCB)

- Uses PRP E (AES), to provide encryption in integrity in one procedure (parallel)
  - Defined over GF(2^{128}), we have 2·D := D·x ∊ GF(2^{128})
  - Pick random IV as usual, let D ← E(K,IV)



OCB (cont’d)

- Checksum for integrity:
  checksum = $m_1 \oplus m_2 \oplus \ldots \oplus m_{last-1} \oplus c_{last}$
- Then ciphertext is
  CT = (IV, c_0, …c_{last}, tag)
- Optional in 802.11i: OCB-AES



D ← 3 D

Checksum

D

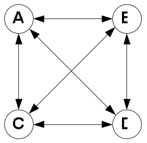
+

E(K,)

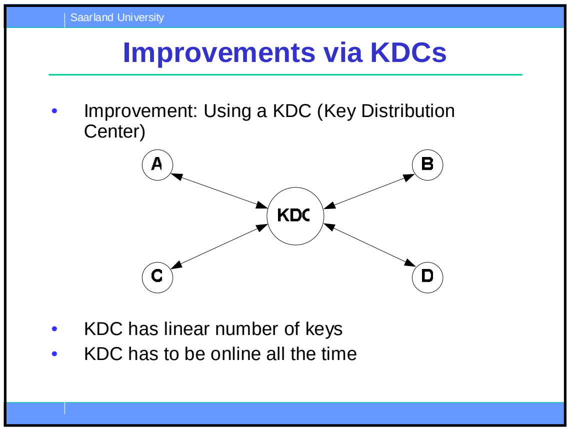
tag

Key Management

- Key management between n parties



- Every party has to manage n keys, n^2 keys overall

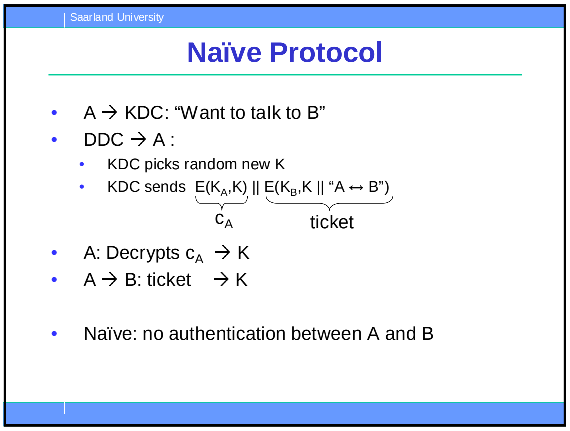


Saarland University

Improvements via KDCs

- Improvement: Using a KDC (Key Distribution Center)

- KDC has linear number of keys
- KDC has to be online all the time



Saarland University

Naïve Protocol

- A → KDC: “Want to talk to B”
- DDC → A :
  - KDC picks random new K
  - KDC sends $\underbrace{E(K_A,K)}_{c_A} \,||\, \underbrace{E(K_B,K \,||\, \text{“}A \leftrightarrow B\text{”})}_{\text{ticket}}$
- A: Decrypts c_A → K
- A → B: ticket → K

- Naïve: no authentication between A and B