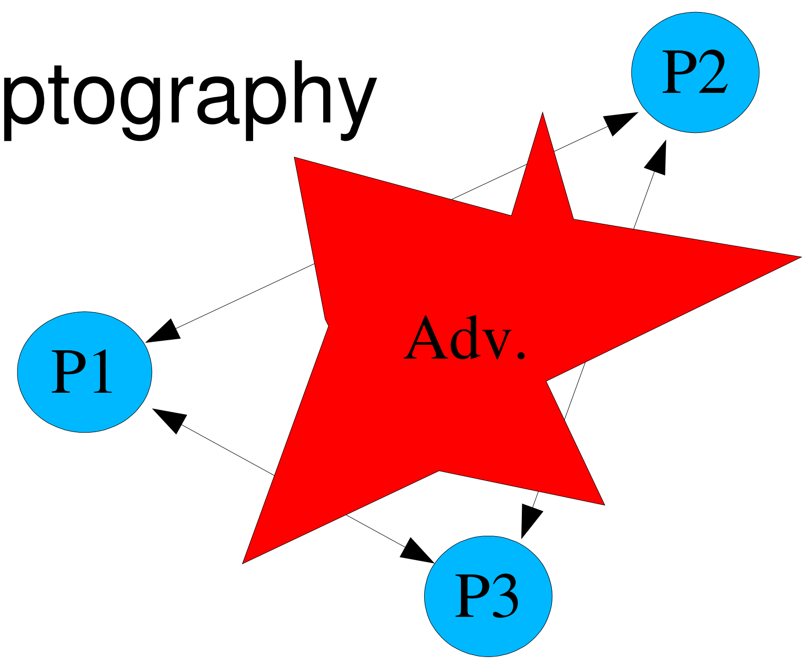
Towards computationally sound symbolic security analysis

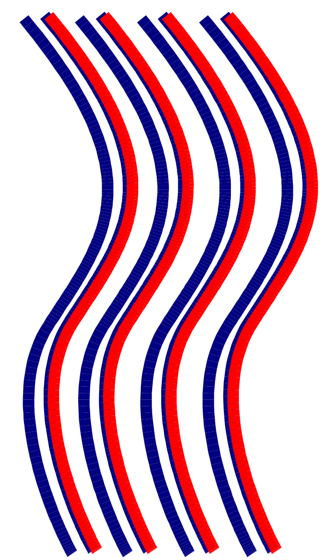
Daniele Micciancio, UCSD

DIMACS Tutorial – June 2004

Security protocols

- Protocols: distributed programs
- Goal: maintain prescribed behavior in adversarial execution environment
- Tool: Cryptography





Analyzing security protocols

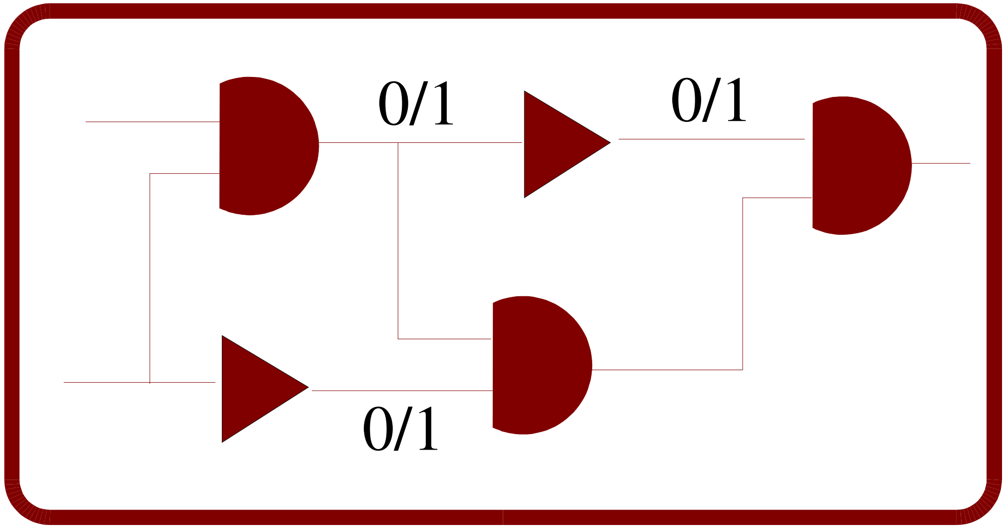
- Typically much more complicated than traditional protocols because of universal quantification over the adversaries
- Implications:
  - Security cannot be tested, but only proved
  - Need for a formal model to precisely formulate and prove security properties

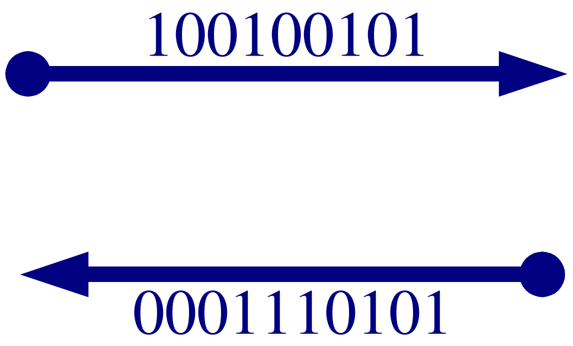
Models of security

- Computational model
  - Encryption [Goldwasser, Micali 1983]
- Symbolic model
  - [Dolev, Yao 1983]
- Other models
  - Random oracle model
  - Generic model

Computational Model

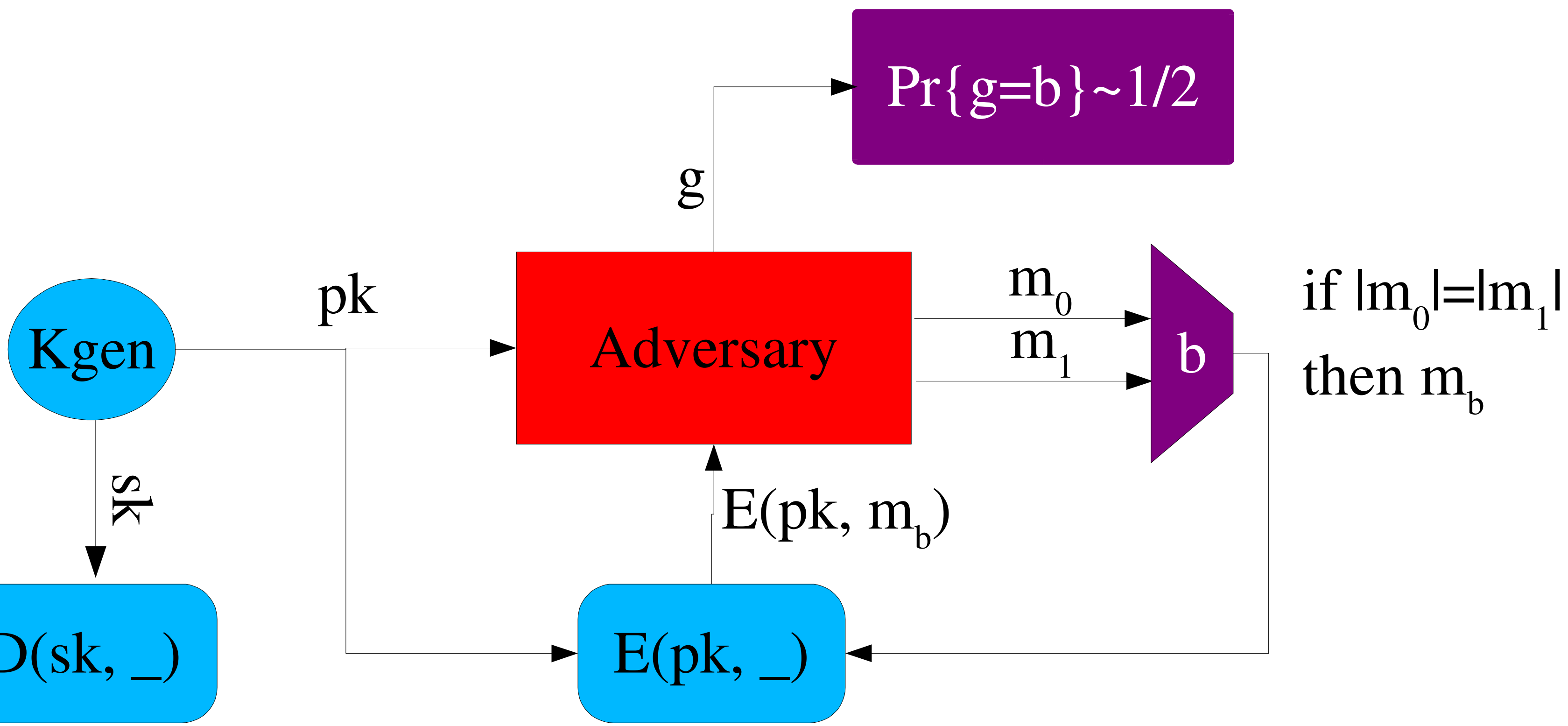
- Detailed model of ***computation*** / *communication*
- Cryptographic operations are <u>*not modeled*</u>, but <u>*defined*</u> within the model.





Example: CPA-secure Encryption

- Encryption scheme = (Kgen, E, D)
- Security against "chosen plaintext attack":



Features of CPA-security

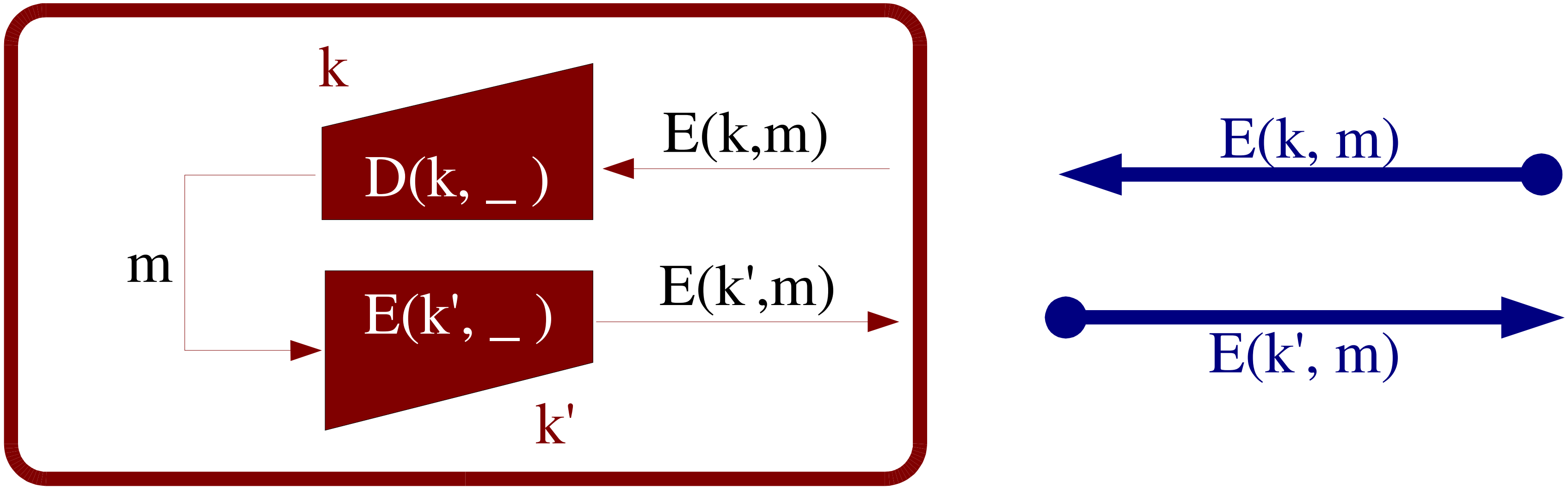
- Even partial information about message is hidden
  - captured by size 2 message space
- No assumption on message distribution
  - captured by adversarially chosen messages
- Strong security (succ. prob. ~ 1/2)
- Encryption function can be used multiple times
  - Letting Adv. make many queries (m_0,m_1) does not make the definition substantially stronger

Non-features of CPA-security

- Message length is not necessarily hidden:
  - Messages must satisfy $|m_0| = |m_1|$
- The key is not necessarily hidden, e.g.:
  - Kgen': Run Kgen->k, and output k' = (k,r)
  - $E'_{(k,r)}(m) = (E_k(m),r)$
- Other definitions are possible:
  - e.g., schemes can completely hide the key

Symbolic model

- Abstract computation and communication model
- Cryptography is integral part of the model: cryptography = abstract data type



Computational model

- Advantages:
  - High security assurance
  - Provides guidance to design of crypto primitives
  - Allows definition of new crypto primitives
- Disadvantages
  - Proofs are long and hard to verify
  - Security intuition is often lost in technical details
  - Few cryptographers still write full proofs, and nobody read them anyway

Symbolic model

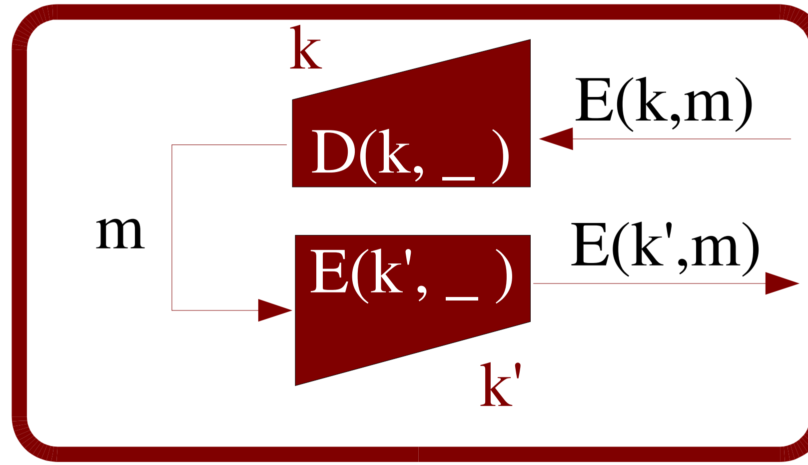
- Potential advantages
  - Simpler, higher level proofs: e.g., no probabilities
  - Automatic proof verification
- Disadvantages
  - Security proved only against abstract adversaries
  - Unclear assumptions on cryptographic primitives
  - Tailored to specific security properties, and classes of protocols

Computational vs. symbolic Adv.

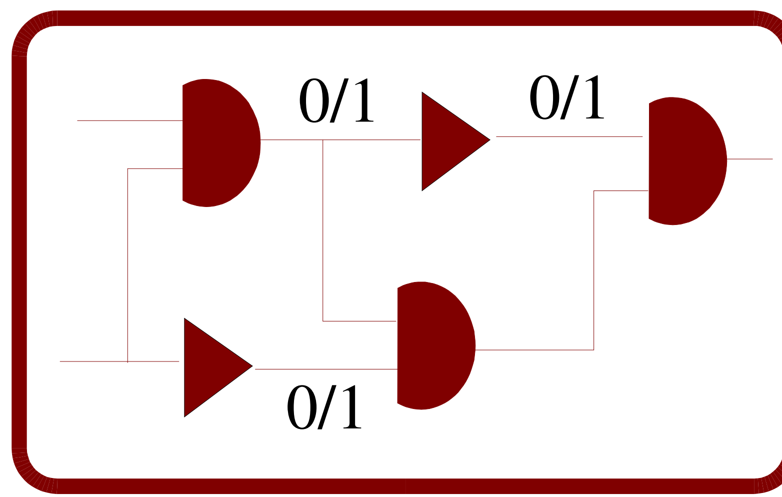
- Computational Adversary:
  - arbitrary probabilistic polynomial time Adv.
  - may break symbolic model assumptions by guessing a key (with non zero probability)
- Symbolic Adversary:
  - restricted but computationally unbounded and/or non-deterministic adversary
  - may break the computational model by non-deterministically guessing a key

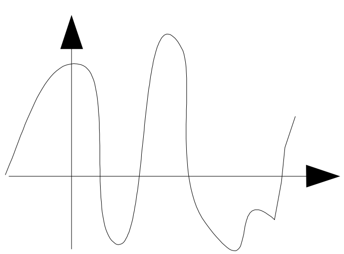
Abstraction Level

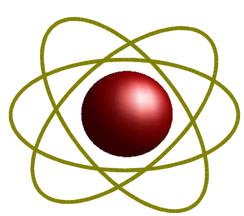
- Security Protocols
- Cryptography



- Digital circuits
- Physics / EE







What level of abstraction should be used to ...

- ... describe security ***protocols***?
  - Higest level that allows to describe the protocol's actions
  - Typically, symbolic model is enough
- ... define security ***properties***?
  - Highest possible that allows to describe all realistic threats (e.g., adversarial's actions)
  - Computational model is typically accepted as a reasonable choice

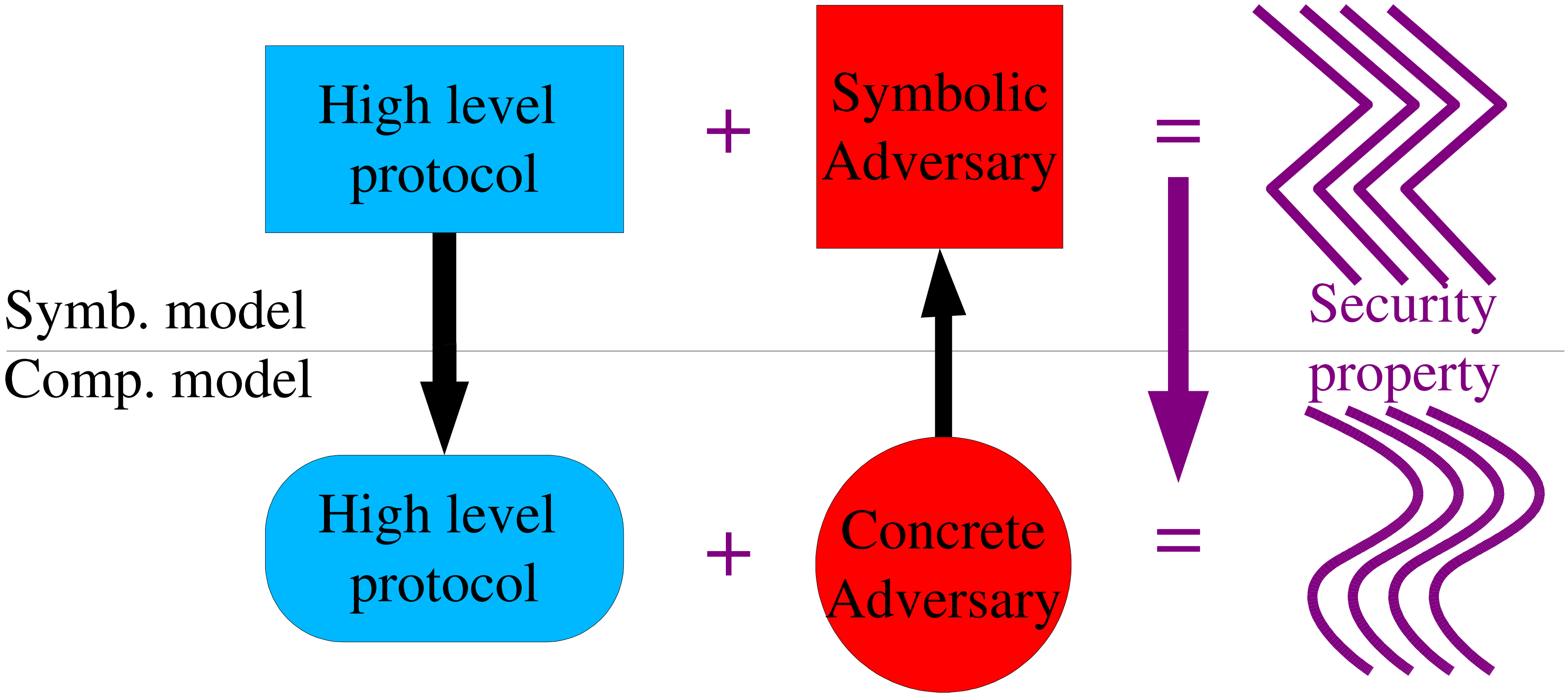
Beyond the computational model

- Power analysis attacks
  - [Kocher]
- Timing attacks
  - [Kocher]
- Sometimes useful:
  - constant round concurrent Zero Knowledge protocols [Dwork, Naor, Sahai] [Goldreich]

Soundness of symbolic analysis

- Goal: framework where
  - protocols are written and analyzed symbolically
  - still, security holds against computational adversaries
- Advantages and limitations
  - Simple protocols and security proofs
  - High security assurance
  - Applies only to a subclass of protocols
  - Targets restricted class of security properties

What is a sound symbolic analysis?



Using the soundness theorem

- High level protocol Prot
- Soundness theorem:
  - For any comp. Adv, if SymbExec[Prot,[Adv]] satisfies S, then CompExec((Prot),Adv) satisfies S
- Symbolic security proof
  - For any symb. Adv', SymbExec[Prot,Adv'] satisfies S
- Strong security guarantee
  - For any comp Adv, CompExec[(Prot),Adv] satisfies S

Remarks

- Standard process in cryptography:
  - E.g. Transformation from semihonest to malicious adversarial models using Zero Knowledge
- Compiling protocols:
  - Usually a non-trivial transformation
  - May introduce inefficiencies (e.g., use of ZK)
- Compiling adversaries:
  - Usually efficiency is not as critical here

What's different with soundness of symbolic analysis?

- Formal high level protocol description language
  - E.g., no probabilities. Important for automation.
- Simple interpretation of high level procols
  - Essential for analysing existing protocols
  - Important for implementation of new protocols
- Compiling adversaries: highly non-trivial
  - Very restricted target language
  - Important for automatic verification

Approaches to sound symbolic analysis

- Secure multiparty computation
  - Library to interpret/compile symbolic programs in computational setting
  - Powerful: Embed symbolic terms in computational model, retaining all capabilities of comp. model
- Ad-hoc approaches
  - Specialized languages for subclasses of protocols
  - Directly justify symbolic analysis

Example: encrypted expressions

- Very simple protocols: “A(input) -> B: output”
- Syntax: X = input | const | $\{X\}_{key}$ | (X,...,X),
- Example: X = (k1, $\{(k3, \{(0, input)\}_{k2})\}_{k1}$, $\{k2\}_{k3}$)
- Computational interpretation [X]:$\{0,1\}^*$->$\{0,1\}^*$
  - Generate keys Kgen->k1,k2,k3
  - Evaluate expression bottom up, where
    - $[\{X\}_k]=E_k([X])$
    - [(X1,...,Xn)] = ([X1],...,[Xn])

Symbolic execution

- On input m, A transmits X' = X[m/input] to B
- The symbolic (Dolev-Yao) adversary, given expression X', computes as much information as possible, according to the following rules:
  - X' is known
  - If (X1,...,Xn) is known, then X1, ..., Xn are known
  - If $\{X\}_k$ and k are known, then X is known

Security properties

- Secrecy of the input:
  - the input value is protected by the protocol
- Computational secrecy:
  - For any input s, the distributions [X](s) and [X](0) are computationally indistinguishable
- Symbolic secrecy:
  - No symbolic (Dolev-Yao) adversary can recover m from X[m/input]

Pattern Semantics

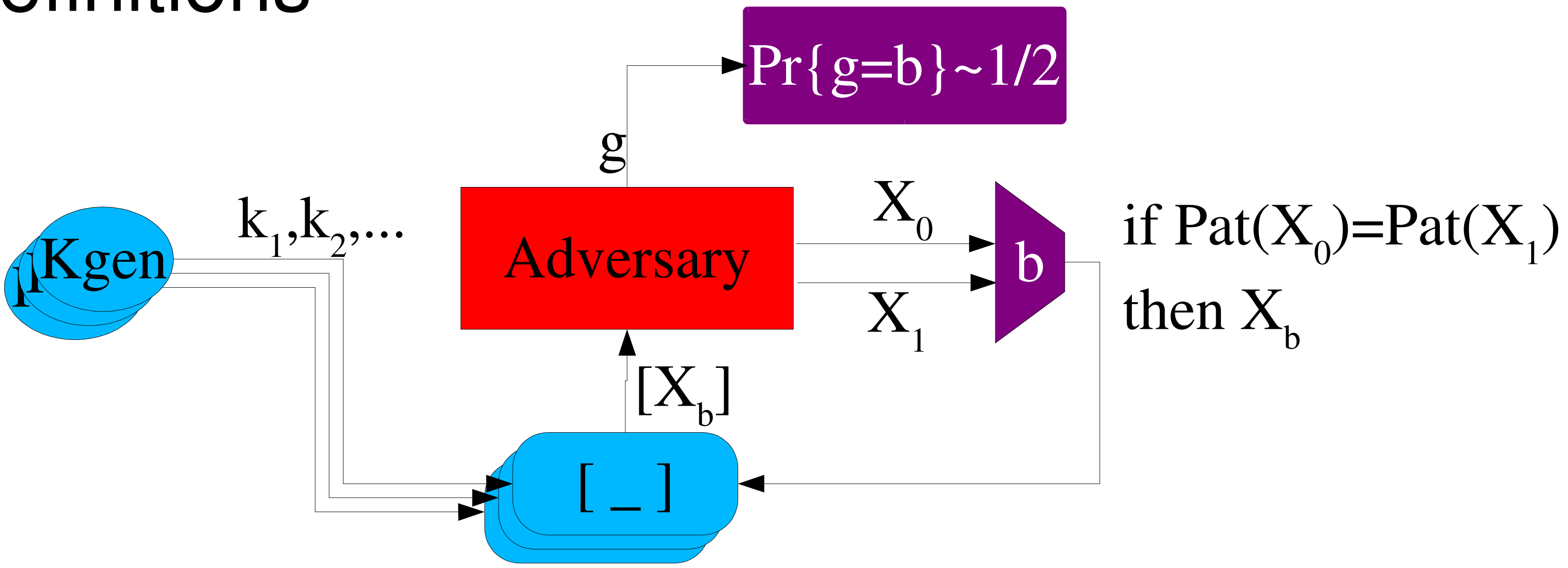
- Associate each program with a pattern:
  - P = input | const | (P,...,P) | $\{P\}_{key}$ | "?"
- Examples:
  - Pattern(k1, $\{(k3, \{(0, \text{input})\}_{k2})\}_{k1}$, $\{k2\}_{k3}$)
    = (k1, $\{(k3, \{(0, \text{input})\}_{k2})\}_{k1}$, $\{k2\}_{k3}$)
  - Pattern(k1, $\{(k3, \{(0, \text{input})\}_{k2})\}_{k1}$, $\{k4\}_{k3}$)
    = (k1, $\{(k3, \text{"?"})\}_{k1}$, $\{k4\}_{k3}$)

Soundness Theorem

- [Abadi-Rogaway] if Pattern(X1)==Pattern(X2) then [X1]~[X2] are computationally indistinguishable, provided that
  - (Kgen, E, D) is “type 0” secure encryption scheme
  - expressions X1, X2 are acyclic, e.g., expression $(\{k1\}_{k2},\{k2\}_{k1})$ is not allowed.
- Corollary:
  - If Pattern(X) does not contain “input”, then X is secure

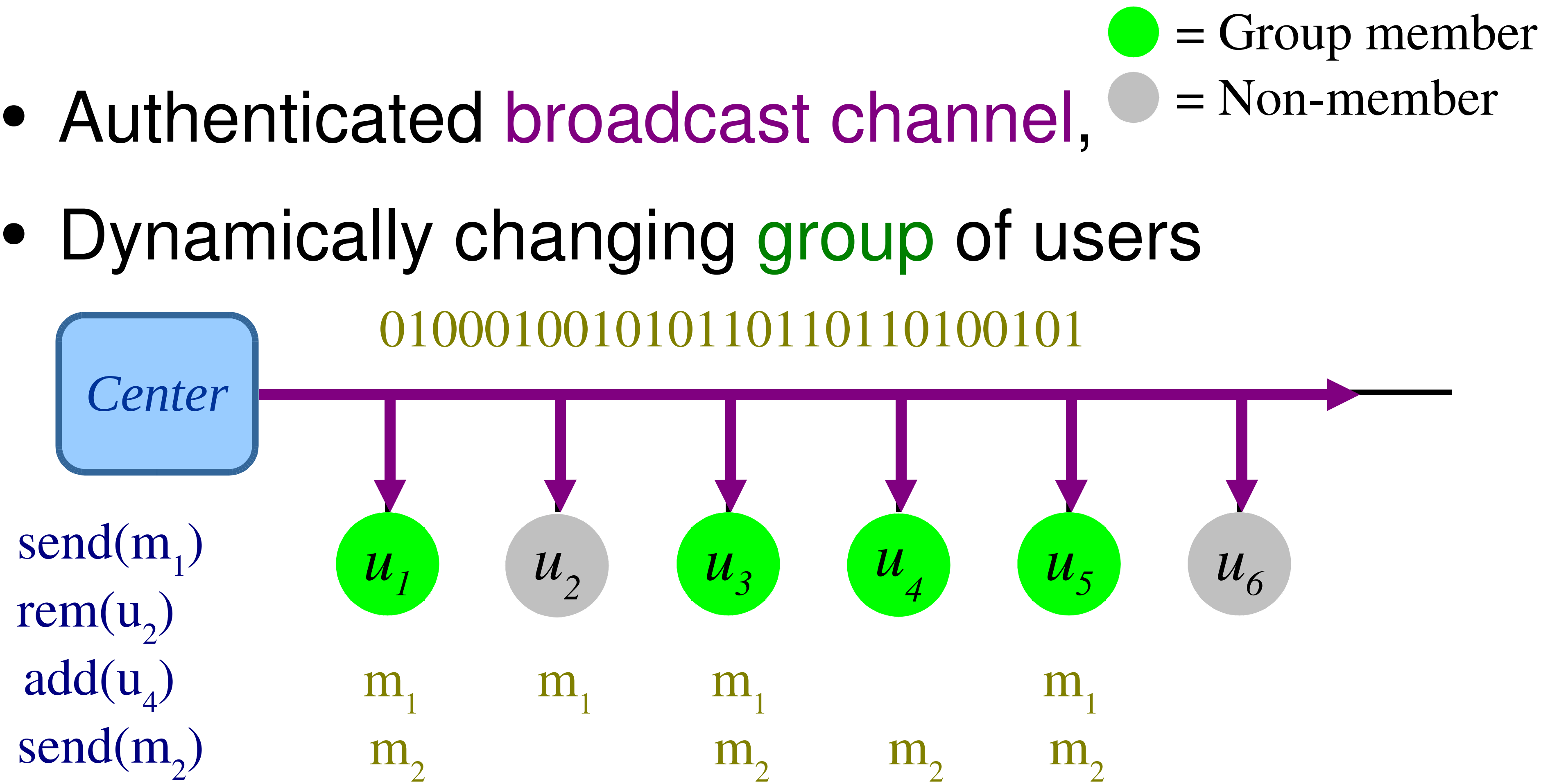
Soundness result as a metatheorem

- Soundness theorem has the form of a standard cryptography result
- As easy to use as normal cryptographic definitions



Case study: Secure multicast

- Authenticated broadcast channel,
- Dynamically changing group of users

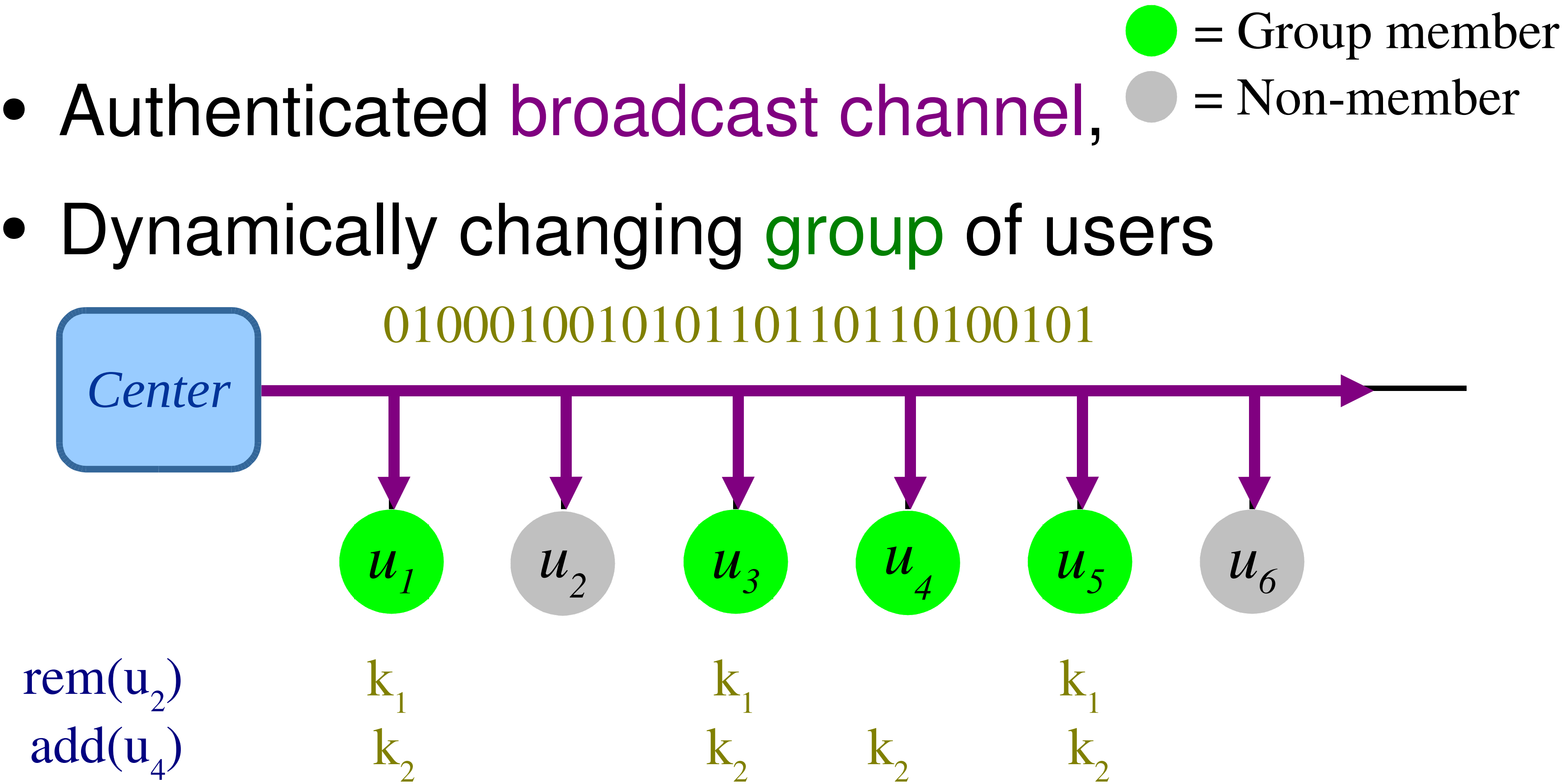


Multicast key distribution problem

- Standard approach to achieve secrecy:
  - Establish a common secret key
  - Use the key to encrypt the messages
- Problem:
  - Update the key when group membership changes
  - Individually sending new key to all members is too expensive
  - Cannot encrypt new key under old one because the old one is compromised

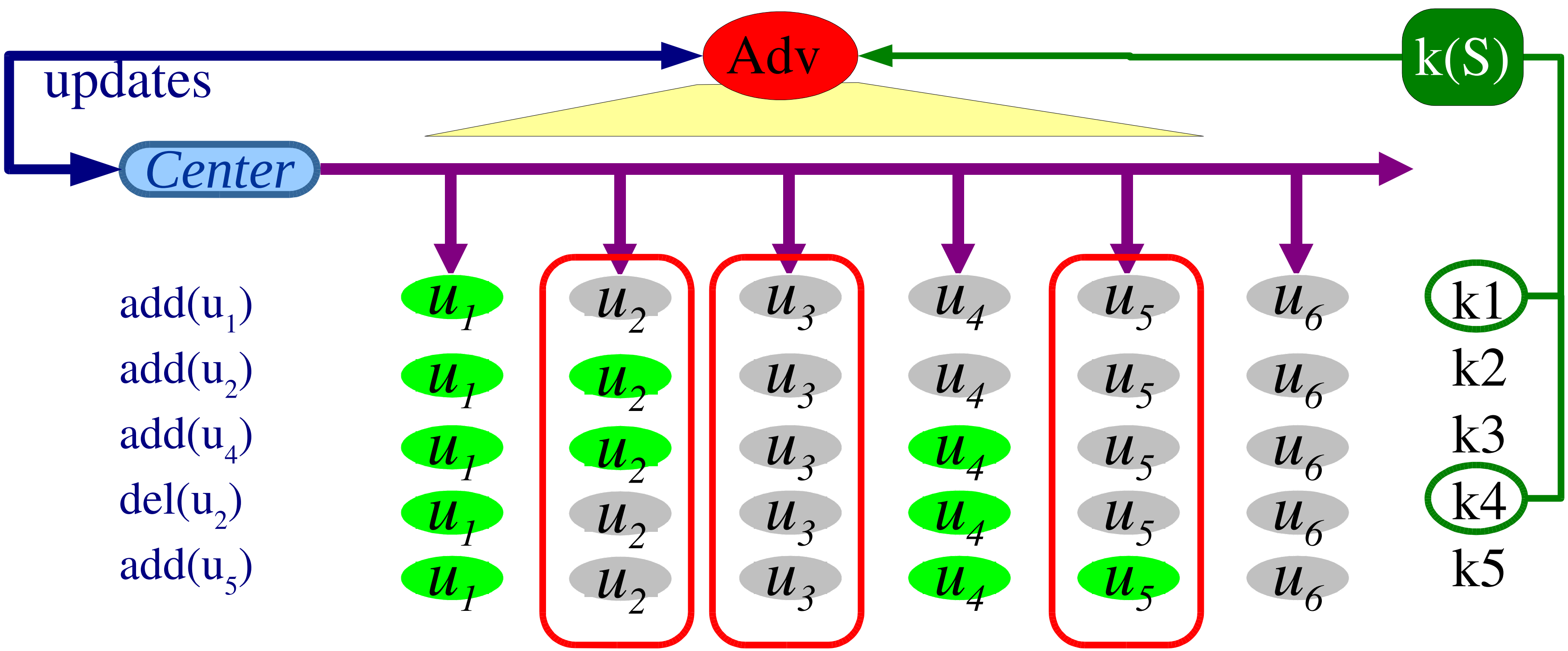
Secure key distribution

- Authenticated broadcast channel,
- Dynamically changing group of users



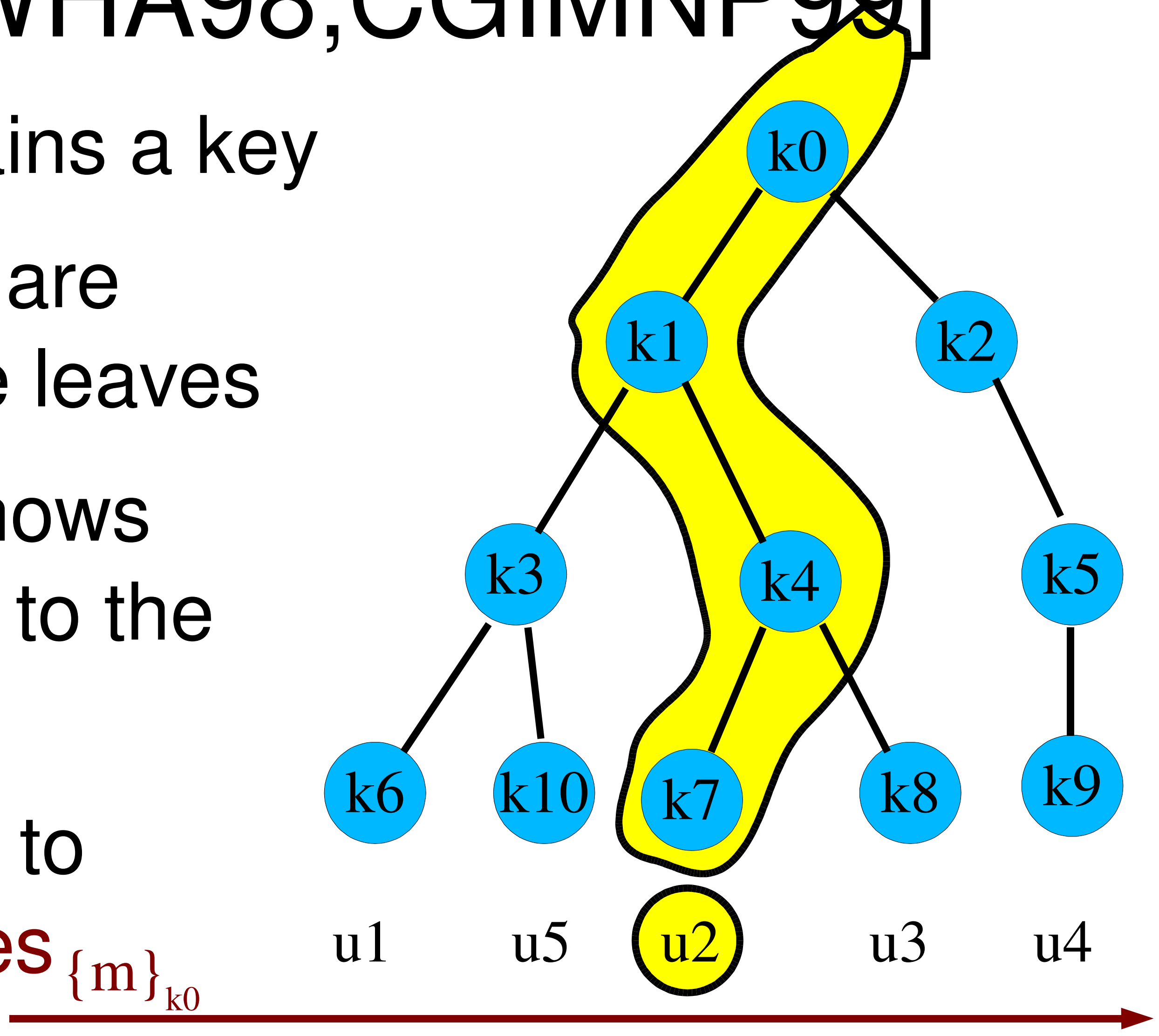
Secure key distribution

- For any sequence of updates, and coalition C, $\{u_C$, xxx, $k(S)\} \sim \{u_C$, xxx, $k'(S)\}$, where S = {t :C does not intersect the group }



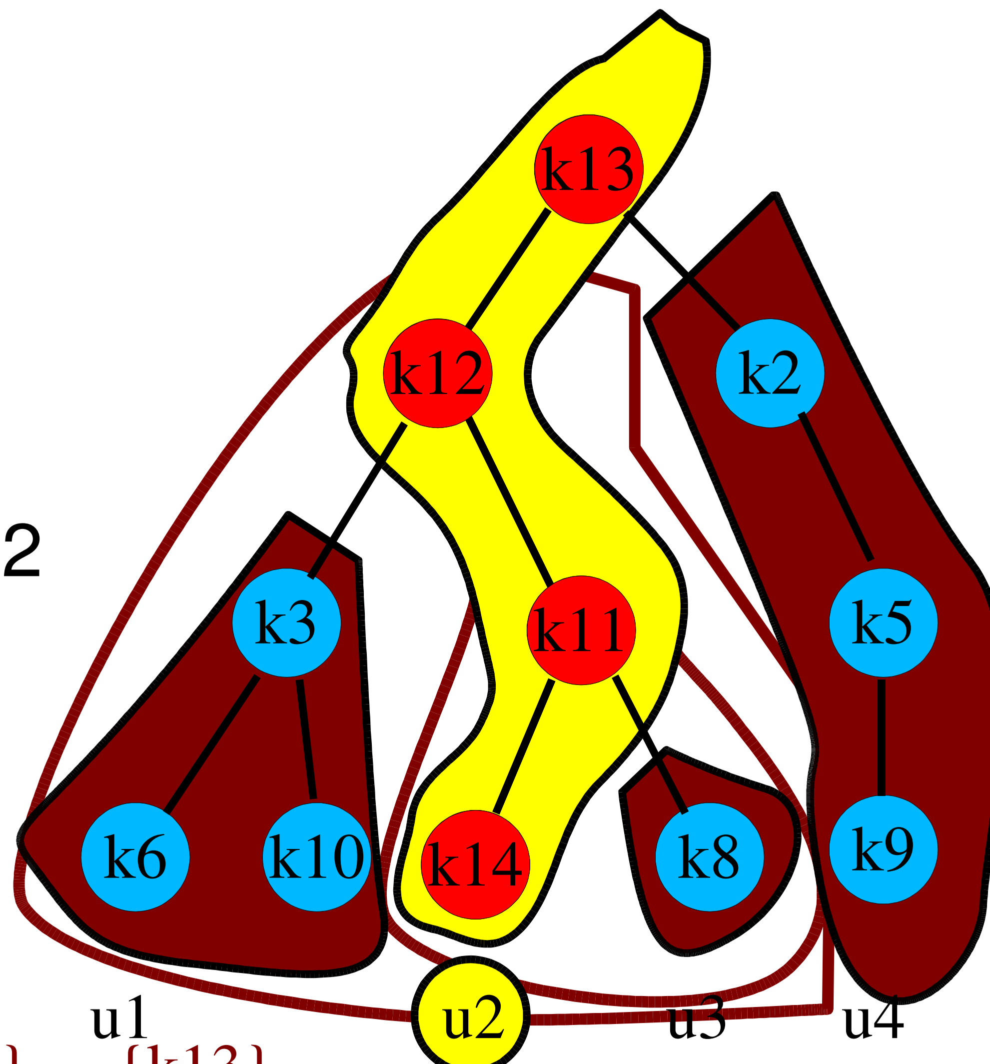
Logical Key Hierarchy [WGL98,WHA98,CGIMNP99]

- Each node contains a key
- Group members are associated to the leaves
- Each member knows keys on the path to the root
- Root key is used to encrypt messages



Updating the group

- E.g., remove u2
- Center sends rekey messages:
  - Change keys known to u2
  - Send each new key to subtrees associated with its children



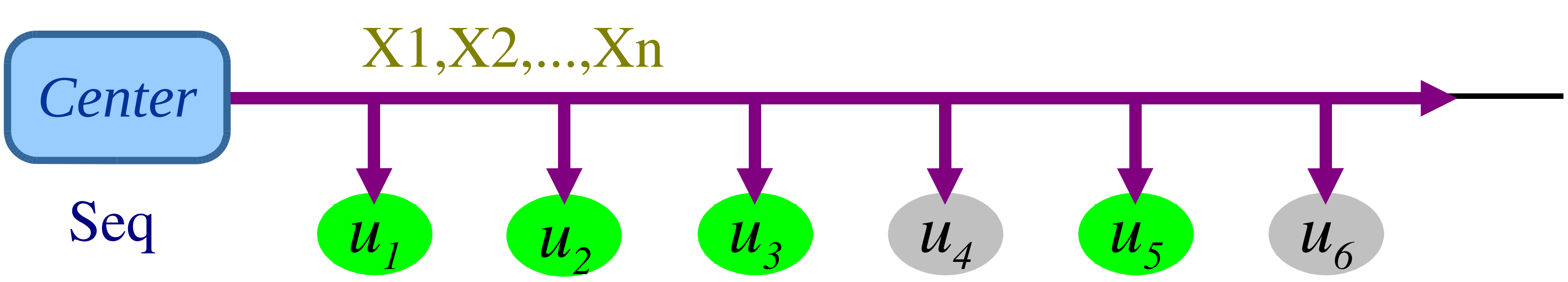
$\{k12\}_{k3}$, $\{k11\}_{k8}$, $\{k13\}_{k2}$, $\{k12\}_{k11}$, $\{k13\}_{k12}$

Abstract key distribution protocols

- Each user has an associated key
- Group center trasmits messages of the form
  - $X = k \mid \{X\}_k \mid (X,...,X)$
- At any given point in time t there exists a key k such that
  - Each group member at time t can recover k
  - Non-members cannot recover k, even if they collude
  - k is not used to encrypt any rekey message

Computational security of multicast key distribution

- Fix a coalition C and a sequence of updates Seq
  - K_S : group keys when none of C is in group
  - No k in K_S can be computed from $(X_1,\ldots,X_{\mathbf{n}})$, U_C
  - keys in K_S are not used to encrypt in $(X_1,\ldots,X_n)$



Computational security of multicast key distribution

- Fix a coalition C and a sequence of updates Seq
  - K_S : group keys when none of C is in group
  - No k in K_S can be computed from $(X_1,\ldots,X_n)$, U_C
  - keys in K_S are not used to encrypt in $(X_1,\ldots,X_n)$
  - K_S is the only occurrence of K_S keys in $\text{Pattern}((X_1,\ldots,X_n),U_C,K_S)$
  - $\text{Pattern}((X_1,\ldots,X_n),U_C,K_S)==\text{Pattern}((X_1,\ldots,X_n),U_C,K'_S)$
  - $[(X_1,\ldots,X_n),U_C,K_S] \sim [(X_1,\ldots,X_n),U_C,K'_S]$

Adversarial updates and corruptions

- We proved that for every sequence of updates Seq and coalition C, the keys K(S) are secure
- What if Seq and C are chosen by the adversary?
  - If Seq and C are chosen at the outset, then security follows from universal quantification
- Can Seq and C be chosen adaptively as the protocol is executed?
  - Definition gets much more complicated

Adaptive adversaries

- Define the following initially empty sets:
  - C = corrupted users
  - K(S) = secure keys
- Adversary can issue the following commands
  - issue a group update operation (add/remove user)
  - if user u was not a member at times t in S: add u to C
  - if none of the member at time t is in C: add t to S
- Polynomial bound on sequence of commands

Is key distribution adaptively secure?

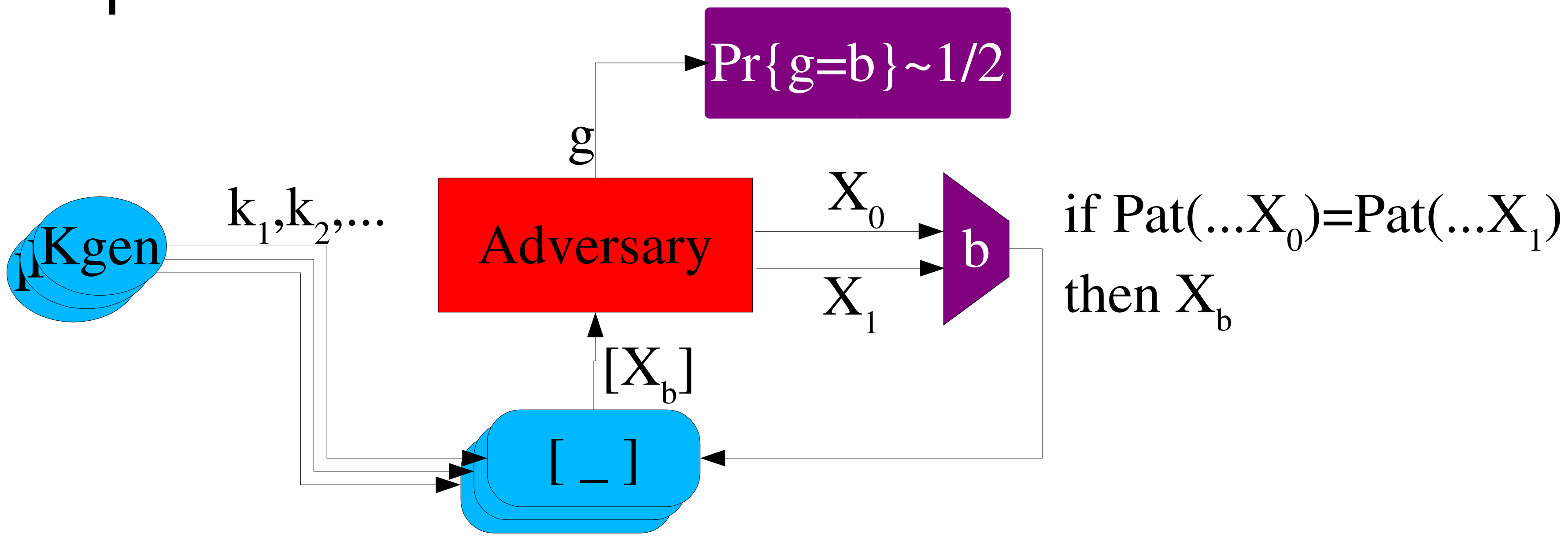
- Symbolic model:
  - A scheme is secure if no adaptive adversary can compute a key in K(S) from messages received during the attack
- Non-adaptive security implies adaptive security:
  - Let Adv be an adaptive adversary
  - Define Seq and C by emulating Adv with protocol
  - Invoke security for every Seq, C, and non-deterministic non-adaptive Adversaries

Is the protocol really secure?

- What about adaptive attacks in the computational setting? Our proof breaks down.
- Problem:
  - Sequence of expressions X1,...,Xn is adaptively chosen, where Xi may depend on [X1], ..., [Xi-1]
  - This allows to define distributions that cannot be expressed as [X]:
  - E.g., Set X1= $\{0\}_k$, X2=b, where b is the last bit of [X1].

Adaptive security of encrypted expressions

- Proving the security of the protocol is related to establishing an adaptive version of the soundness theorem for encrypted expressions:



Selective decommitment/decryption

- Consider the following adaptive adversary:
  - X1 = ($\{m1\}_{k1}$, $\{m2\}_{k2}$, ..., $\{mn\}_{kn}$)
  - X2 = (ki: for a random subset of the i's)
- Question: are the mj (for kj not in X2) still secret?
  - Standard hybrid arguments break down
- Classic open problem in cryptography
  - Byzantine agreement (early 80's)
  - [Dwork,Naor,Reingold,Stockmeyer 03]

Some extensions to the AR logic

- Completeness:
  - [X1] = [X2] => pattern(X1) = pattern(X2) ?
  - [Micciancio,Warinschi02/04] No under [AR] assumptions. Yes if authenticated encryption is used.
  - [Gligor,Horvitz03] same under weaker assumptions
- Realistic encryption functions:
  - What if encryption reveals the length of the message?
  - [MW02/04] Refine logic with patterns “?”n
- Abadi-Jurens: security against passive attacks

Dealing with message lengths and encryption keys: a new semantics

- Structure of expressions:
  - Struct(k) = key; Struct(c) = const
  - Struct(X1,...,Xn) = (Struct(X1),...,Struct(Xn))
  - Struct($\{X\}_k$) = {Struct(X)}
- Pattern(X) = Pat(X,Keys(X))
  - Pat(k,K) = k; Pat(c,K) = c,
  - Pat((X1,...,Xn), K) = (Pat(X1,K),...,Pat(Xn,K))
  - Pat($\{X\}_k$,K) = $\{\text{Pat}(X,K)\}_k$ if k is in K
  - Pat($\{X\}_k$,K) = $\{\text{Struct}(X)\}_k$, if k is not in K

Claims about new Pattern Semantics

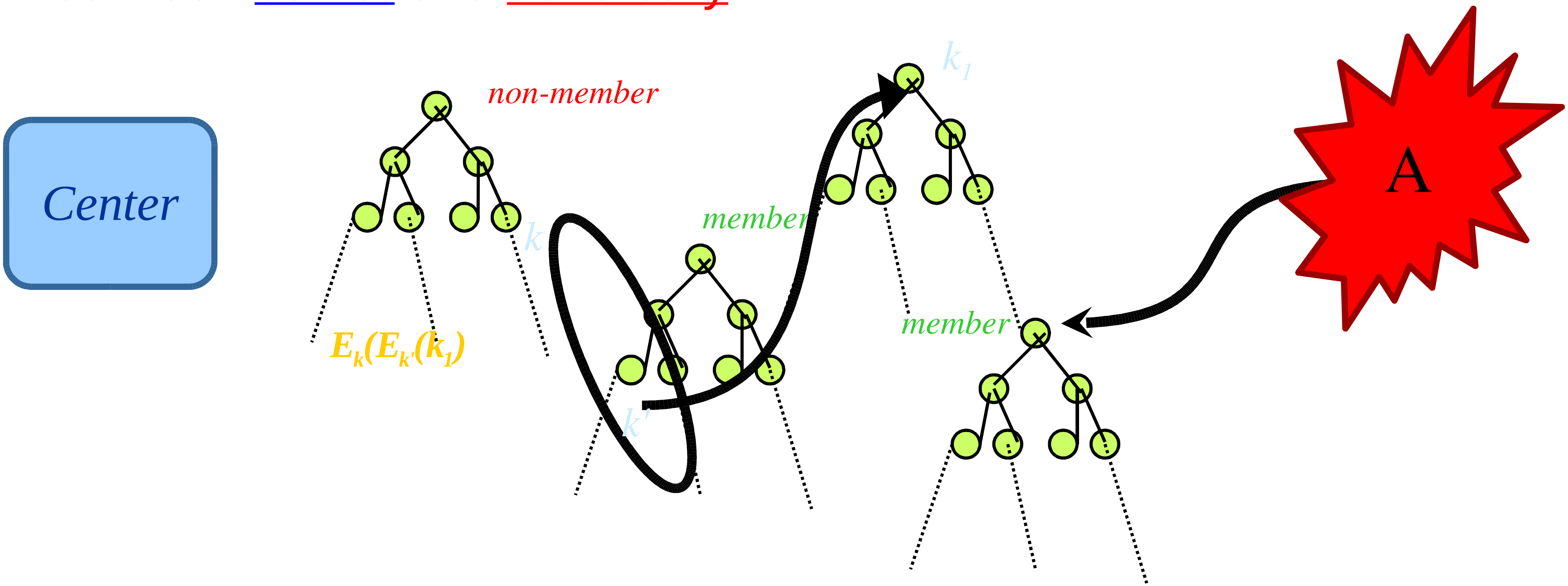
- Claim 1: New notion suffices in most application
  - it seems a good security practice anyway
- Claim 2: For any CPA secure encryption,
  - if Pattern(X1) = Pattern(X2) then [X1]~[X2]
- Claim 3: If Pattern(X1)=/=Pattern(X2) then
  - there is a CPA encryption such that [X1]~/~[X2]

Other applications

- Symbolic model can be used not only to analyse security, but also to prove lower bounds
- [Micciancio,Panjwani04]: O(log n) communication lower bound
  - Protocols may use pseudo random generators arbitrarily nested with encryption operations
  - Symbolic attacks can be easily translated into computational ones
  - If replace operation is allowed, constant in O(log n) matches best protocol in the model [CGIMNP99]

Micciancio-Panjwani: proof idea

- View a multicast key distribution protocol as a game played between center and adversary.



- Adversary changes labels on the keys which are labeled *member* or *non-member*.
- Center introduces rekey messages, modeled as hyper-edges over the keys.

Other extensions

- What if the adversary can alter/inject packets?
- Recent work on active attacks:
  - [Micciancio,Warinschi 04] : CCA / trace properties
  - [Laud 04] : CPA+ / secrecy properties
  - [Bakes,Pfitzman 04] : Compiler / multiparty computation
- Selective decommitment issue

Open problems: formal methods

- Extend with other cryptographic primitives:
  - PRGs, PRFs, Hash, Signatures, etc.
- Extend to universal composability setting, etc.
- Foundamental questions in basic setting:
  - Find most general conditions under which adaptive soundness of encrypted expressions can be proved
  - Develop formal methodsds / tools for the automatic analysis of multicast key distribution protocols

Open problems: cryptography

- Find encryption scheme (e.g., Cramer-Shoup) such that soundness of encrypted expressions holds without the acyclicity restriction
- Find encryption scheme such that adaptive soundness of encrypted expressions holds without any syntactic restriction

Conclusion

- There is not a single “right” security model
- Multiple computational security definitions:
  - CPA, CCA, authenticated encryption, etc.
  - => Several corresponding symbolic models
- Symbolic model should allow to specify simple and clear computational security properties
- Plenty of work for everybody
  - Automation, security modeling, protocol design, etc.