Applied Cryptography and Network Security

Adam J. Lee
adamlee@cs.pitt.edu
6111 Sennott Square

Lecture #9: Threshold Cryptography
February 4, 2014

University of Pittsburgh

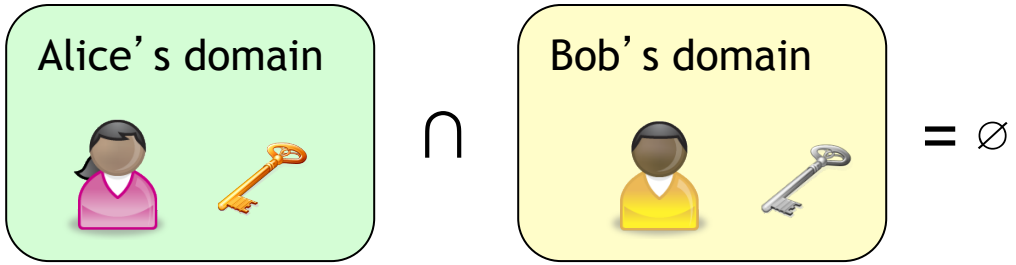
Announcements

1. HW #1 will be posted by Thursday
   - Details on the web page
   - Due on Thursday February 20

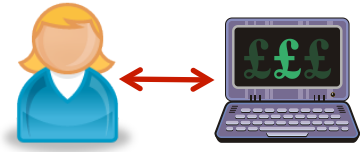
2. Project #2 due Sunday February 9

3. Check the webpage for an announcement re: project submission instructions

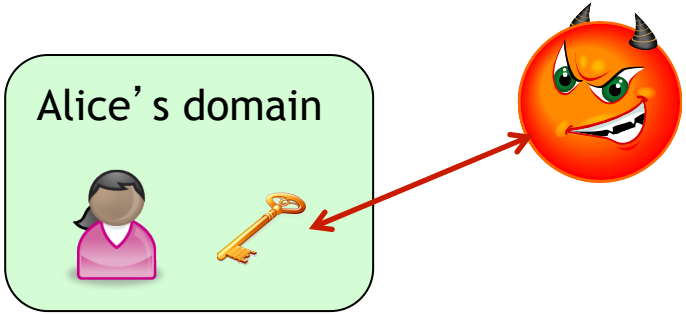
To date, we've (explicitly and implicitly) made a number of assumptions when discussion cryptosystems



Key Ownership



Device Utilization



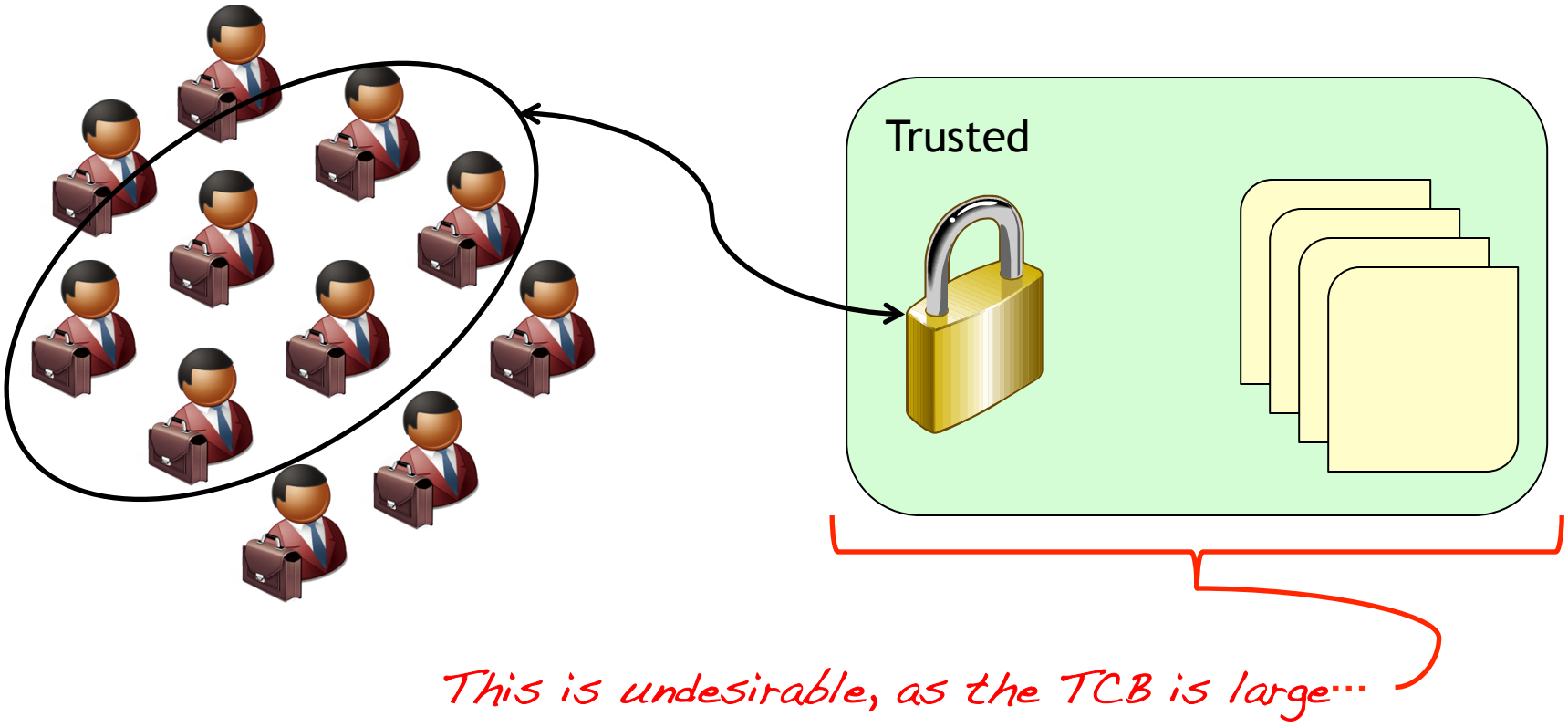
Key Compromise

We've seen several cryptosystems that meet our needs relative to these assumptions

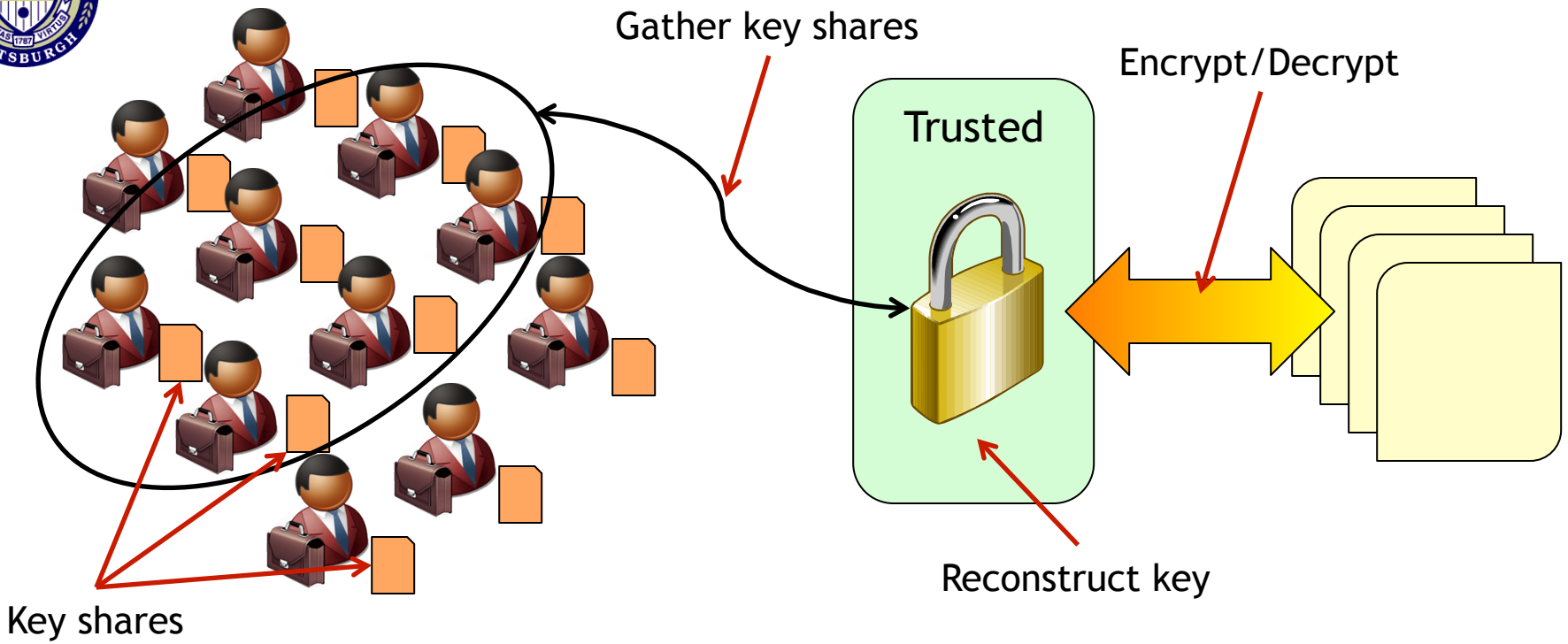
What happens if these assumptions are violated?

A motivating example…

Scenario: Eleven scientists are working together on an extremely sensitive joint project. To ensure that results are not tampered with, at least six scientists (i.e., a quorum) must be present in order to read or alter experimental results.



A better solution involves using a shared secret key



Goal: Given n users and a threshold k < n, divide a secret D into n pieces D_1, D_2, …, D_n such that:

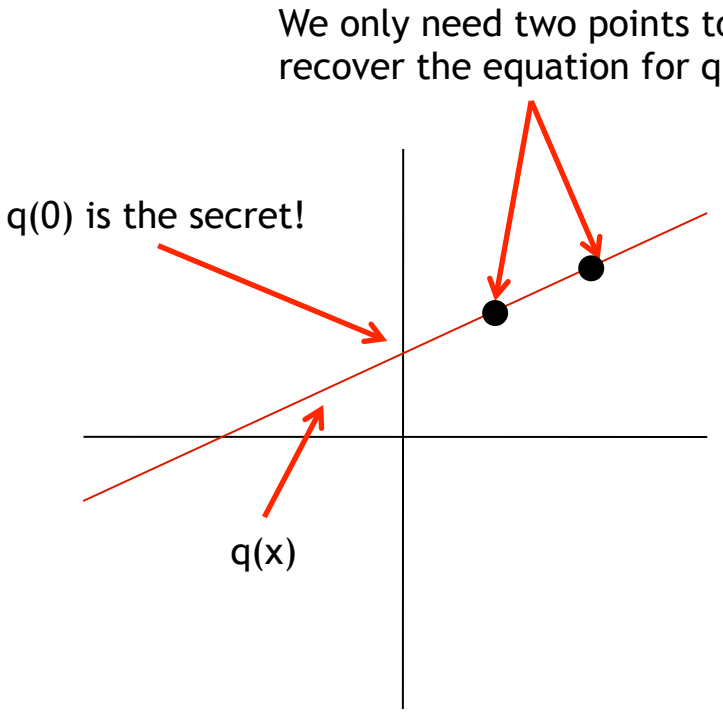
1. Any group of k or more users can jointly obtain the secret
2. Any group of k-1 or less users cannot obtain any information about the secret (i.e., all possible secrets remain equally likely)

Shamir's (k, n) secret sharing system allows us to accomplish this!

Paper: Adi Shamir, "How to Share a Secret," Communications of the ACM 22(11): 612-613, November 1979.

Intuitively, this scheme is quite simple

- Choose a random (k-1)-degree polynomial q(x) whose root is D
- Each key share is a point on q(x)
  - $D_1 = q(1)$
  - $D_2 = q(2)$
  - …
  - $D_n = q(n)$
- Distribute $\{D_1, …, D_n\}$ to participants
- q(x) can be recovered by interpolation using any k shares
- q(0) = D



Toy Example: A line defines a (2, n) secret sharing scheme

Details…

Rather than using real arithmetic, use modular arithmetic

- E.g., The set of integers modulo a large prime *p*, $\mathbf{Z}_p$
- Note that *p* must be larger than both D and n

Otherwise we can't represent D!

Otherwise we will have collisions in our shares!

How do we randomly generate a (k-1)-degree polynomial q(x) such that q(0) = D?

- Note that $q(x) = D + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$ is such a polynomial
- So, we just need to choose $a_1, \ldots, a_{k-1}$ at random!
- More precisely, choose $a_1, \ldots, a_{k-1}$ from the uniform distribution [0, p)

Note: All D_i values are computed modulo *p*

How is the secret recovered?

To recover D from a set of shares $\{D_1, \ldots, D_k\}$, solve the following system of k equations with k unknowns:

- $q(1) = D + a_1 1 + a_2 1^2 + \ldots + a_{k-1} 1^{k-1} = D_1$
- $q(2) = D + a_1 2 + a_2 2^2 + \ldots + a_{k-1} 2^{k-1} = D_2$
- …
- $q(k) = D + a_1 k + a_2 k^2 + \ldots + a_{k-1} k^{k-1} = D_k$

Unknowns: $D, a_1, \ldots, a_{k-1}$

Note that any k shares can be used, not just $\{D_1, \ldots, D_k\}$

There exist $O(n \log^2 n)$ algorithms for polynomial interpolation

- In other words, the complexity of this operation is minimal
- For most (k, n) schemes, even a quadratic algorithm is fine

Is Shamir's scheme secure?

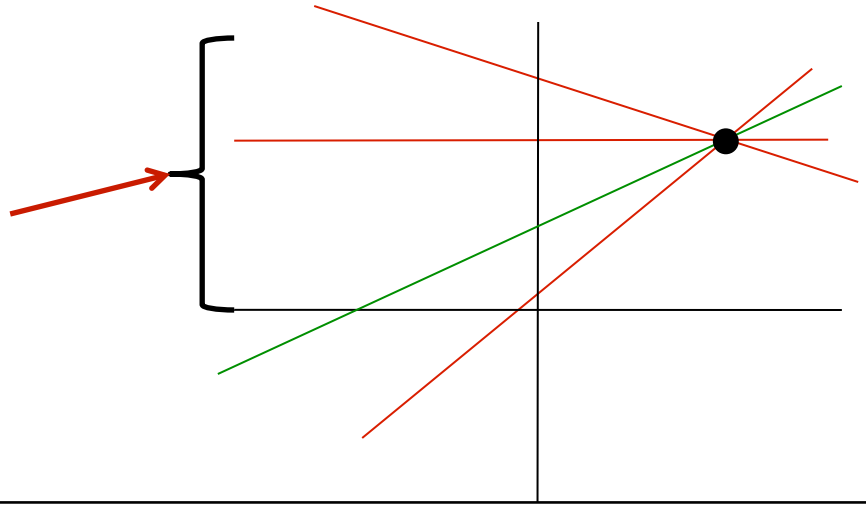
Recall: To prove security, we must show that any attacker who has k-1 shares thinks that all possible values of D are equally likely

Assume that the attacker has k-1 shares D_i

- For each candidate secret $D_c \in [0, p)$, exactly one k-1 degree polynomial $q'(x)$ can be constructed such that $q'(0) = D_c$ and $q'(i) = D_i$ for each known share
- So, all p possible polynomials are equally likely!

Degenerate Example: (2, n) secret sharing

This entire space can be covered!



Shamir's scheme has many nice properties

The size of each share is less than or equal to the size of the secret D

- Why?

For a fixed k, shares can be dynamically added or deleted without affecting other shares!

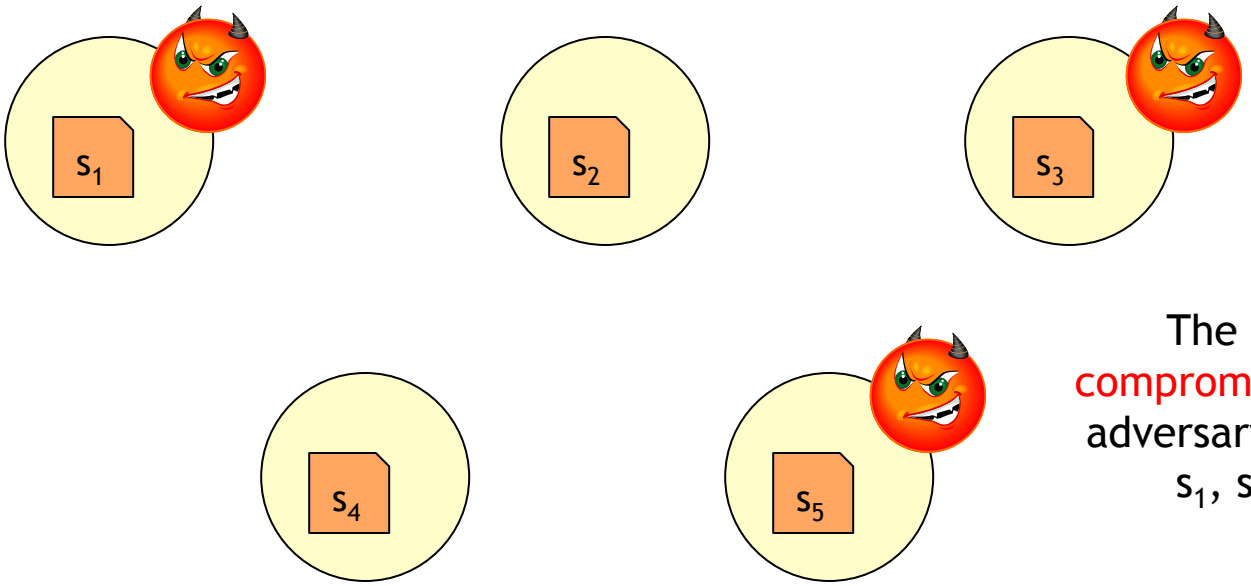
- Adding is easy---just compute new points along q(x)
- How do we delete a share?!?

We can develop a hierarchical scheme where the number of shares needed to find D depends upon the importance of principals

- ***Example:*** (3, n) scheme
- 3 shares to president, 2 shares to each VP, 1 share to each board member
- Who can recover the secret?

This scheme is a building block that allows us to tradeoff security and reliability according to the values of k and n chosen

What happens as machines are compromised over time?



The system is compromised since the adversary has learned s_1, s_3, and s_5!

Example: A (3, 5) secret sharing system

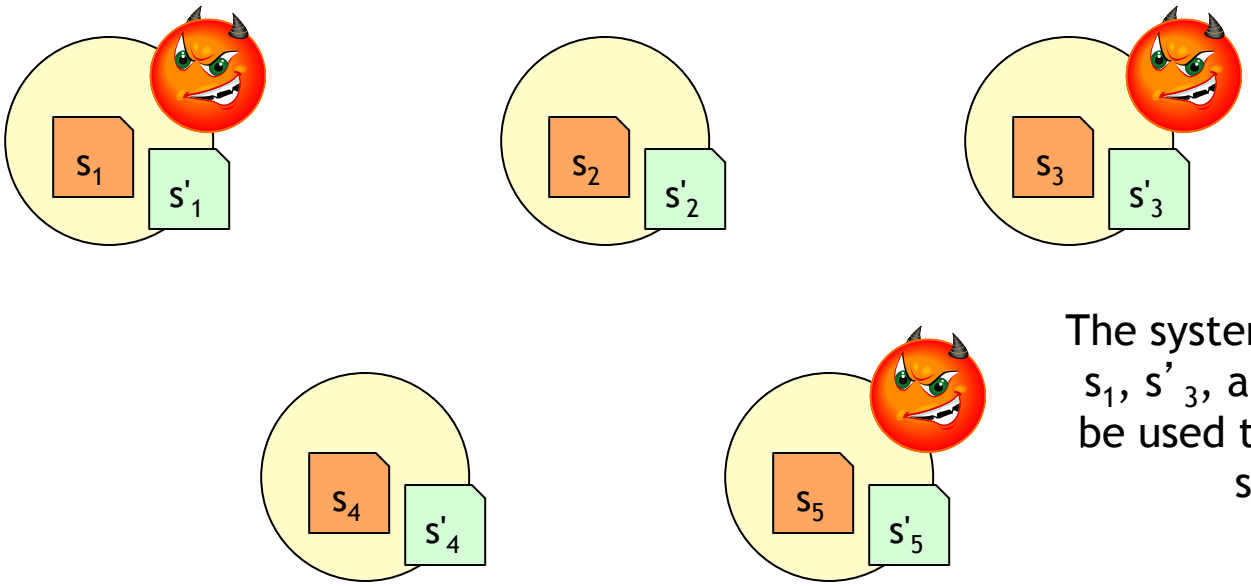
Consider a mobile adversary, capable of compromising nodes

- Over time, more and more nodes compromised
- The adversary learns all secrets stored on each compromised node

If 3 nodes are compromised, the secret is leaked!

- Note: Compromises do not need to be simultaneous!

Proactive secret sharing solves this problem!



The system is safe since s_1, s'_3, and s'_5 cannot be used to recover the secret!

Assumptions:

- Time proceeds in epochs (hours, days, weeks, etc)
- At the start of each epoch, machines can be rebooted into a safe state

Main idea:

- Update secret shares at the start of each epoch
- New shares should invalidate old shares!
- Secret should not change (despite new shares)
- Shares from new epoch cannot be used in conjunction with old shares

How can we do this?

During the setup phase, a trusted dealer

- Determines q(x) and shares as in Shamir's scheme
- Shares sent out to participants

To evolve shares, each participant P_i

- Creates a random (k-1)-degree polynomial $\delta_i(x)$ s.t., $\delta_i(0) = 0$
- Send $u_{ij} = \delta_i(j) \pmod q$ to each P_j
- After receiving all u_{ji}, set $D_i' = D_i + (u_{1i} + \ldots + u_{ni}) \pmod q$

This share is independent of the old share

Note: After share evolution $q'(x) = q(x) + \delta(x)$

- $\delta(x) = \delta_1(x) + \ldots + \delta_n(x)$
- Since $\delta_i(0) = 0$ for each i, $\delta(0) = 0$
- So $q'(0) = q(0) + \delta(0) = D + 0 = D$

The secret is unchanged, despite new shares!

Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung, "Proactive Secret Sharing or: How to Cope with Perpetual Leakage," CRYPTO 1995.

What are the properties of this PSS scheme?

The authors define security in terms of two important properties

Robustness: The new shares computed at the end of the update phase correspond to the secret D (i.e., any subset of k new shares can be used to interpolate the secret D).

Informally, we get this because $\delta(0) = 0$

Secrecy: An adversary that at any time period knows no more than k-1 shares learns nothing about the secret.

Note that old shares become obsolete, since from the adversary's perspective, all possible update polynomials are equally likely. Then, within a single epoch, the proof follows from the secrecy of the Shamir scheme.

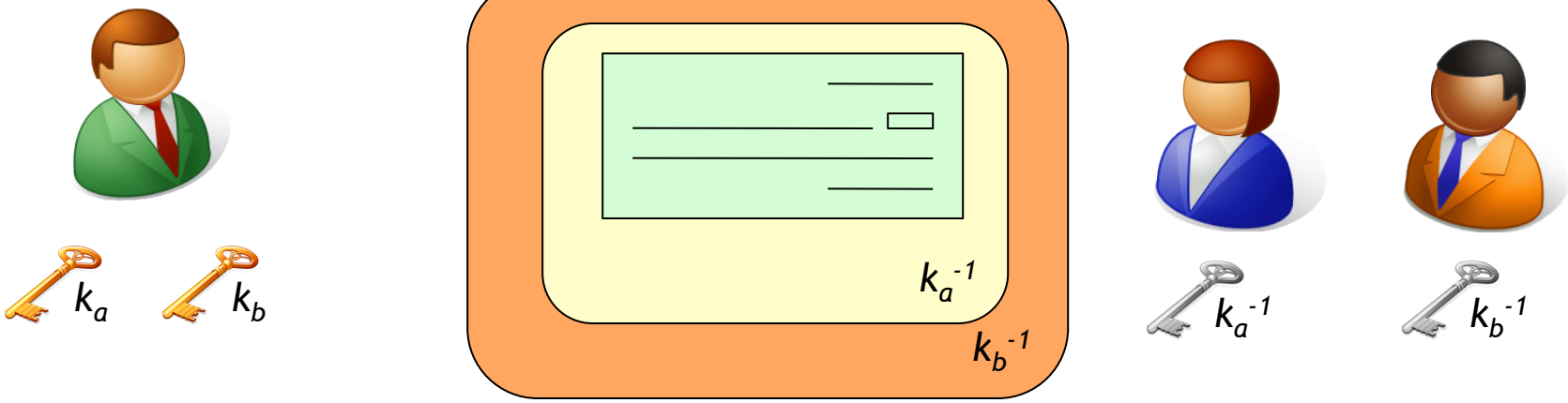
Certain distributed computing problems cannot be solved by using shared symmetric keys

Problem 1: Suppose that a company issues electronic checks, but requires the signatures of at least 3 executives to be considered valid.

Problem 2: A user wants to send an encrypted message to an organization. However, for accountability, the users wants at least 2 members of that organization to view the message together.

Solving these types of problems requires a public key cryptosystem where encryption and/or decryption requires cooperation!

A Straw-Man Approach



Require multiple digital signatures on the message!

Unfortunately, this solution has several pitfalls

- Verifying k signatures is a waste of time
- Message size is proportional to k
- Does not allow for group anonymity
- Doesn't work for decryption case

A better solution would be to define threshold variants of cryptosystems such as RSA

Main idea: Split the RSA decryption exponent *d* into shares using Shamir's secret sharing approach

How does the protocol work?

- Given a message m, each signer can compute a partial signature of m using his or her share of the decryption exponent
- These partial signatures can be multiplied together
- *Result*: A signature on m

This protocol has a rigorous proof of security based upon standard cryptographic hardness assumptions

Want more details? See: Victor Shoup, "Practical Threshold Signatures," Proceedings of EuroCrypt 2000.

So… In what applications would someone actually use all of this stuff?

One Application: An online certificate authority

A certificate authority is responsible for managing the bindings between users (or services) and their public keys

This is typically done offline to protect the private signing key of the CA

This entails three main jobs

- Verifying user identity and/or attributes
- Issuing certificates attesting to that identity or those attributes
- Managing lists of revoked or expired certificates

Ideally, this requires each request/response to be digitally signed

Challenge: How can we make a secure online CA?

Lidong Zhou, Fred B. Schneider, and Robbert van Renesse. COCA: A Secure Distributed On-line Certification Authority. ACM Transactions on Computer Systems 20(4) : 329-368, November 2002.

COCA uses a set of very weak system assumptions in order to define a very resilient online CA

Assumption 1: Byzantine Failures

- An adversary can compromise servers that implement the service
- Compromised servers can behave arbitrarily and/or disclose any local information (including secrets) stored on that server
- An adversary can only compromise fewer than 1/3 of all the servers in any given epoch

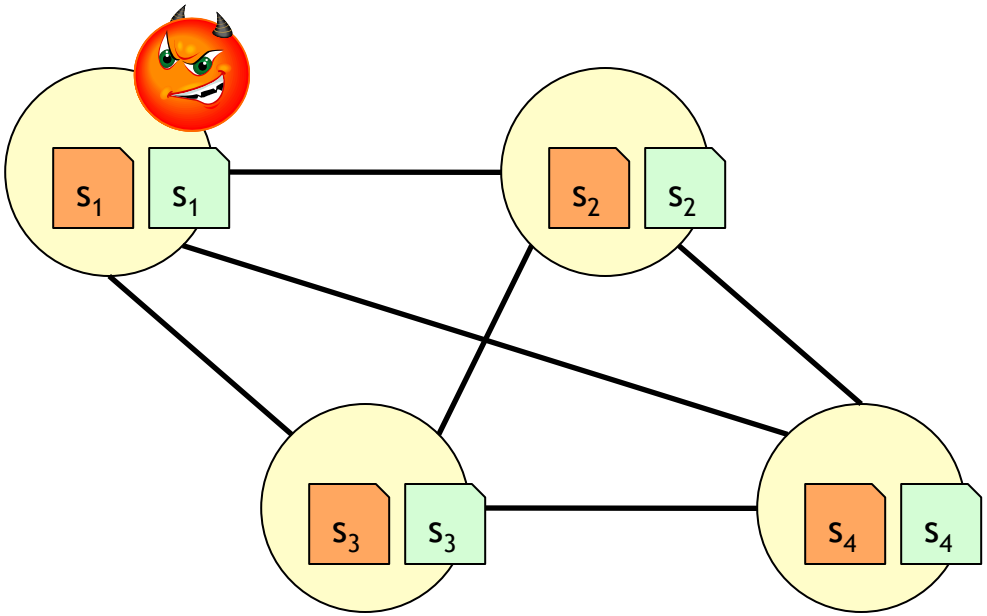
Assumption 2: Active Link Attacks

- An adversary can insert, modify, replay, and delete messages

Assumption 3: Asynchronous Systems

- There is no bound on message processing time or transmission delay
- DoS attacks can delay messages or slow servers by arbitrary finite amounts

So how does COCA work?



COCA uses an interesting combination of cryptographic and distributed systems techniques

- Replication with Byzantine quorum
  - i.e., Group "votes" on actions to take
- Threshold cryptography to protect private signing key from compromise
- Proactive secret sharing to defend against a mobile adversary
- Other DoS resistance techniques

Despite all of these protections, COCA maintains reasonable performance

Check certificate validity

Table III. Performance of COCA Over the Internet[a]

COCA Operation	Mean (msec)	Std dev. (msec)
Query	2270	340
Update	3710	440
PSS	5200	620

[a]The averages and sample standard deviations are from 100 repeated executions during a three-day period.

Evolve secret shares

Create new certificate

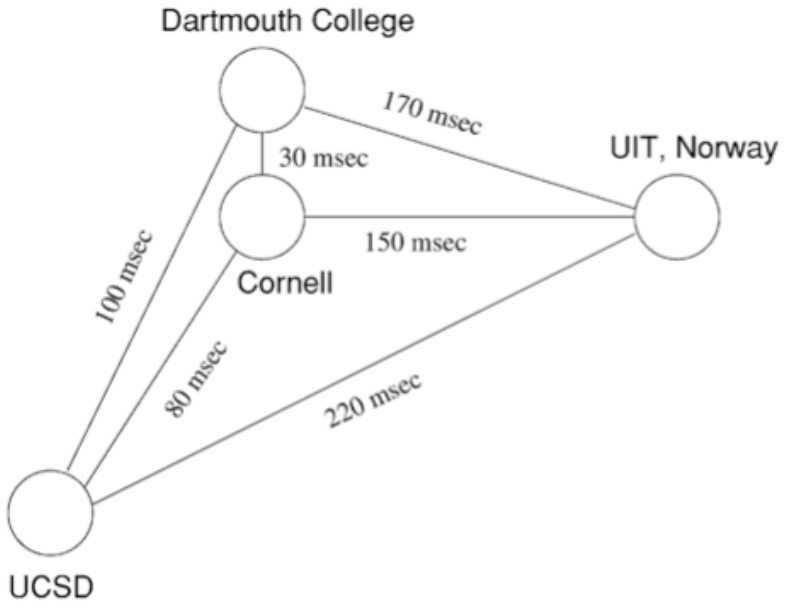


Table IV. Breakdown of Costs for Query, Update, and Proactive Secret-Sharing (PSS) in Internet Deployment

	Query (%)	Update (%)	PSS (%)
Partial Signature	8.0	8.7	
Message Signing	3.2	2.5	2.6
One-Way Function			7.8
SSL			1.6
Idle	88.0	87.7	87.4
Other	0.8	1.1	0.6

Most time is spent waiting on quorum protocols

Conclusions

Threshold systems are useful when trying to secure distributed systems

Interesting properties:

- Knowing k shares allows a secret to be reconstructed
- Knowing even k-1 shares provides no information

Extensions to this basic model have been proposed to allow

- Share evolution
- Tolerance of an untrusted "dealer"
- Shared signatures
- …

Next time: Authentication and identity