Authenticating Cloud Storage with Distributed Keys

Jason Resch

Cleversafe

Presentation Agenda

- The mission of cloud storage
- Properties of existing authentication techniques
  - Password, Token, PKI
- Challenges in securing access to cloud resources
- Conception of a new authentication technique:
  - Distributed Keys
- Implementation and Analysis
- Applications

The Mission of Cloud Storage

- Cloud storage presents unique challenges:
  - Users expect flexible access from any location
  - Many nodes are involved in storing the data
  - The system must be able to scale indefinitely
    - Requires decentralization of critical services
    - Decentralization eliminates single points of failure
- Can these goals be met without sacrificing security?

Existing Authentication Mechanisms

- Password
  - HTTP Digest, Basic
  - LDAP simple bind
- Token
  - Kerberos
- Public Key Infrastructure (PKI)
  - TLS and SSL
  - SSH

Password Authentication

- Advantages
  - Easy to deploy, support and use
  - No special hardware required
- Disadvantages
  - P or D(P) disclosed for each authentication
  - P or D(P) exists somewhere outside user's mind
  - If authentication servers are offline, system fails
  - Computers getting better at breaking passwords
    - Human memory isn't getting better at recalling them!

Password Reuse

- **Password reuse is a growing problem:**
  - Average Internet user has dozens of accounts with different service providers
    - Social networking, e-mail, shopping, online banking, etc.
  - Its nearly impossible for humans to memorize a unique password for each site
    - Consequently passwords are reused
  - When reused, a breach of one is a breach of all
    - Likelihood that one's password is exposed increases linearly with the number of accounts the user has

In the news…

- **Recent cases:**
  - In 2010, the blog network Gawker was compromised, exposing the passwords of 1.3 million users; forced to register to post comments
  - In 2011, the open source project hosting site SourceForge was attacked, affecting the security of over 2 million user accounts
  - In 2011, 10 million users of the mobile application Trapster were notified that their e-mail address and password were compromised

But aren't passwords hashed?

- Salting and hashing passwords stored on the authentication server is a good practice, but it doesn't solve the problem:
  - The average 8 character password has just 18 bits of entropy, can be brute forced in 2^{18} attempts.
  - Average PayPal password found to have 40 bits
  - GPUs can test billions of password per second
- Prudence requires a compromise of a password database be treated as an exposure of the passwords

Kerberos Authentication

- Advantages
  - Breach of server does not yield user's credential
  - Password not sent over the network
- Disadvantages
  - Breach of authentication server yields all accounts
    - It is not possible to even use hashing on the server
  - Attacker watching the network can conduct an offline attack against the user's credential.
  - Requires centralized management, does not scale

Public Key Authentication

- Advantages
  - Credentials not disclosed during authentication
  - Credential not known by any other entity
  - Availability does not depend on a central service
- Disadvantages
  - Users need to carry key with them in hardware
    - Subject to physical theft and loss
  - Special key cards and readers are typical
  - More difficult to deploy and support

Authenticating to the Cloud

- Implementers of cloud storage are forced to choose between several sub-optimal authentication systems:
  - A system whose security is inversely proportional to the number of nodes in the cloud
  - A system with poor availability and scalability
  - A system that is inconvenient and hard to use
- At my company, we were faced with this dilemma:
  - How can we make the authentication system reliable without sacrificing security?

The Solution: Distributed Keys

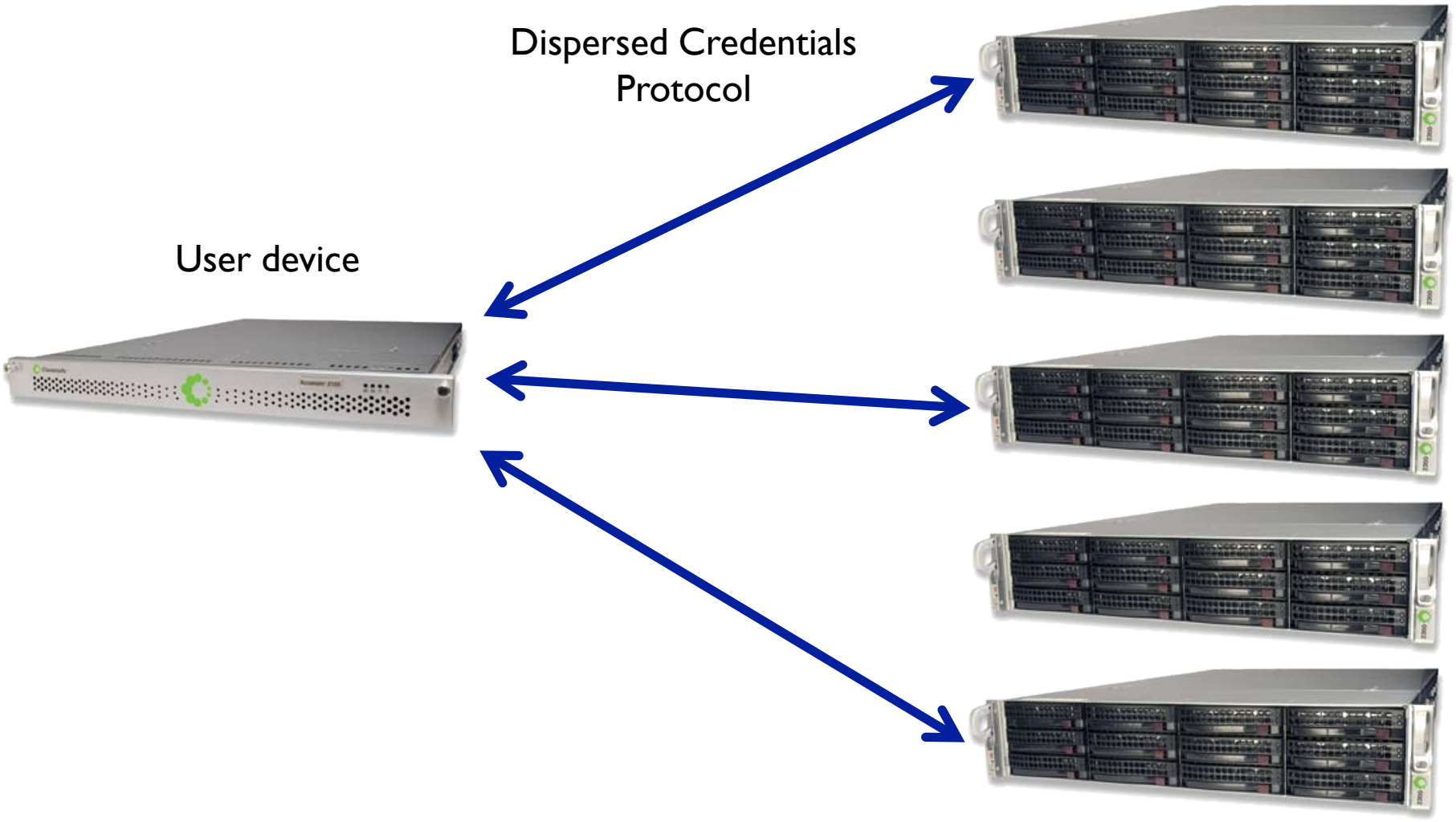
- Distributed Keys enable end users to recover a private key from any location on the network
  - It bridges the gap between password authentication and PKI authentication
    - Seems like password authentication to end users
    - Seems like PKI authentication to service providers
- Unlike more naïve approaches, nothing enabling an offline attack exists at any location
  - Breach of authentication server yields nothing!

Distributed Keys Architecture

User device

username:	jsmith01
password:	********

Distributed Keys Architecture

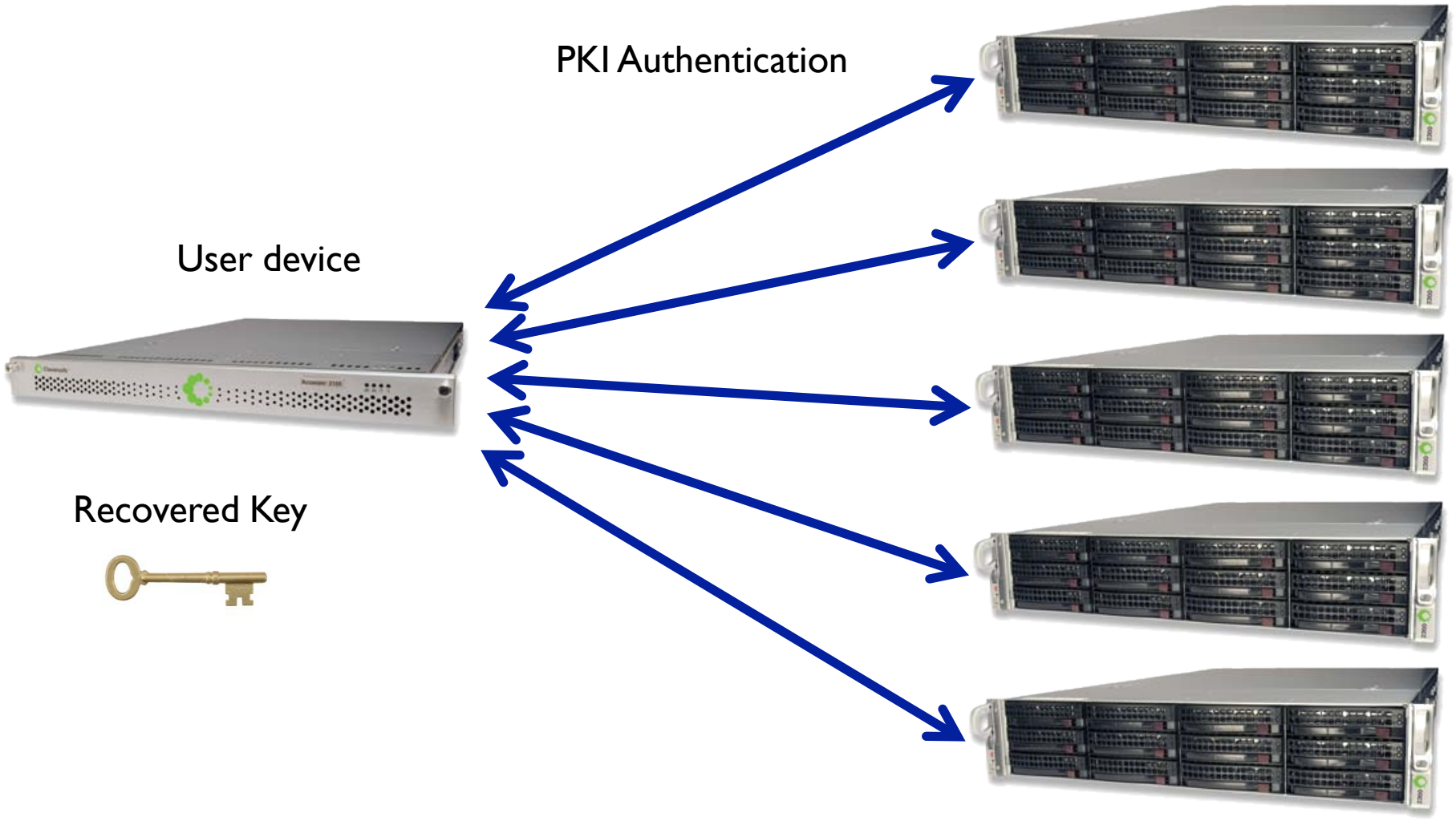


Distributed Keys Architecture

User device

Recovered Key

Distributed Keys Architecture



Distributed Keys Architecture

User device

Recovered Key

Comparison of Mechanisms

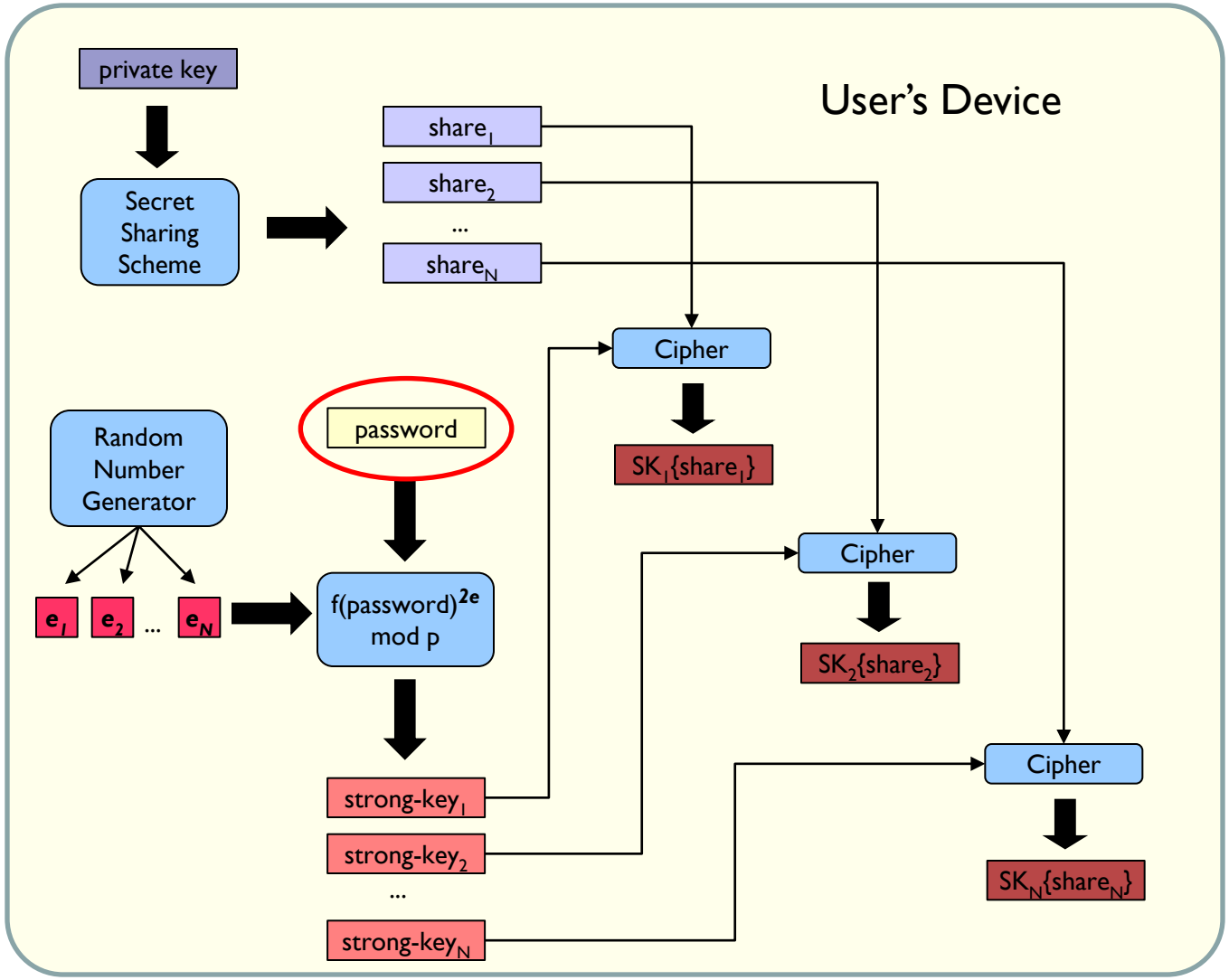
	Password	PKI	DK
1. No single point of failure	✗	✓	✓
2. No single point of compromise	✗	✗	✓
3. Enables access from any location	✓	✗	✓
4. Easy to use	✓	✗	✓
5. Immune to offline brute-force attacks	✗	✗	✓ *
6. Credentials are not disclosed during use	✗	✓	✓
7. Immune to physical theft	✓	✗	✓

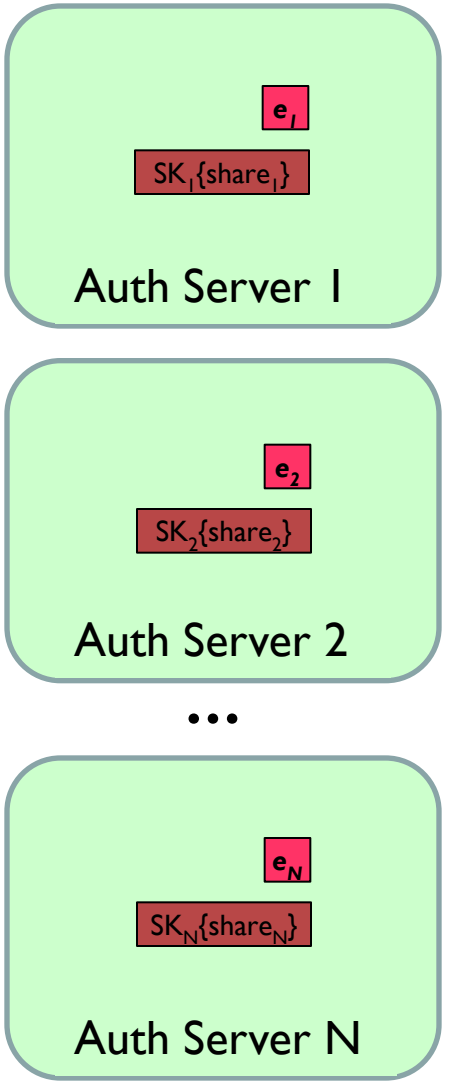
* Requires a threshold number of simultaneous compromises

How it Works

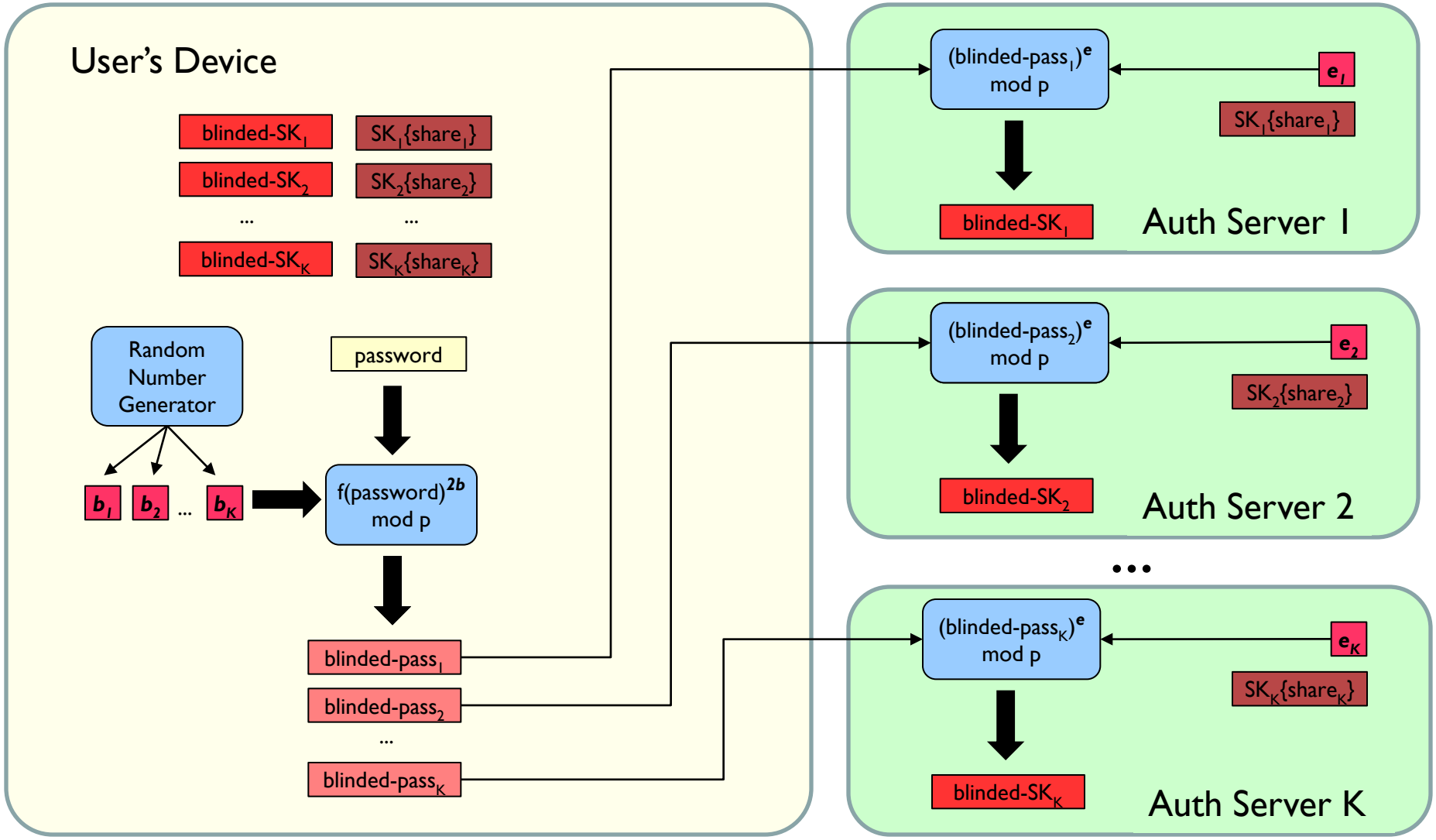
- We found that through a combination of various cryptographic protocols, an authentication system with almost ideal properties could be formed
  - Server-assisted strong secret generation
    - Warwick Ford and Burton S. Kaliski Jr. (2000)
  - Secret Sharing
    - Adi Shamir and George Blakley (1979)
  - Encryption and Digital Signatures

Distributed Key Storage

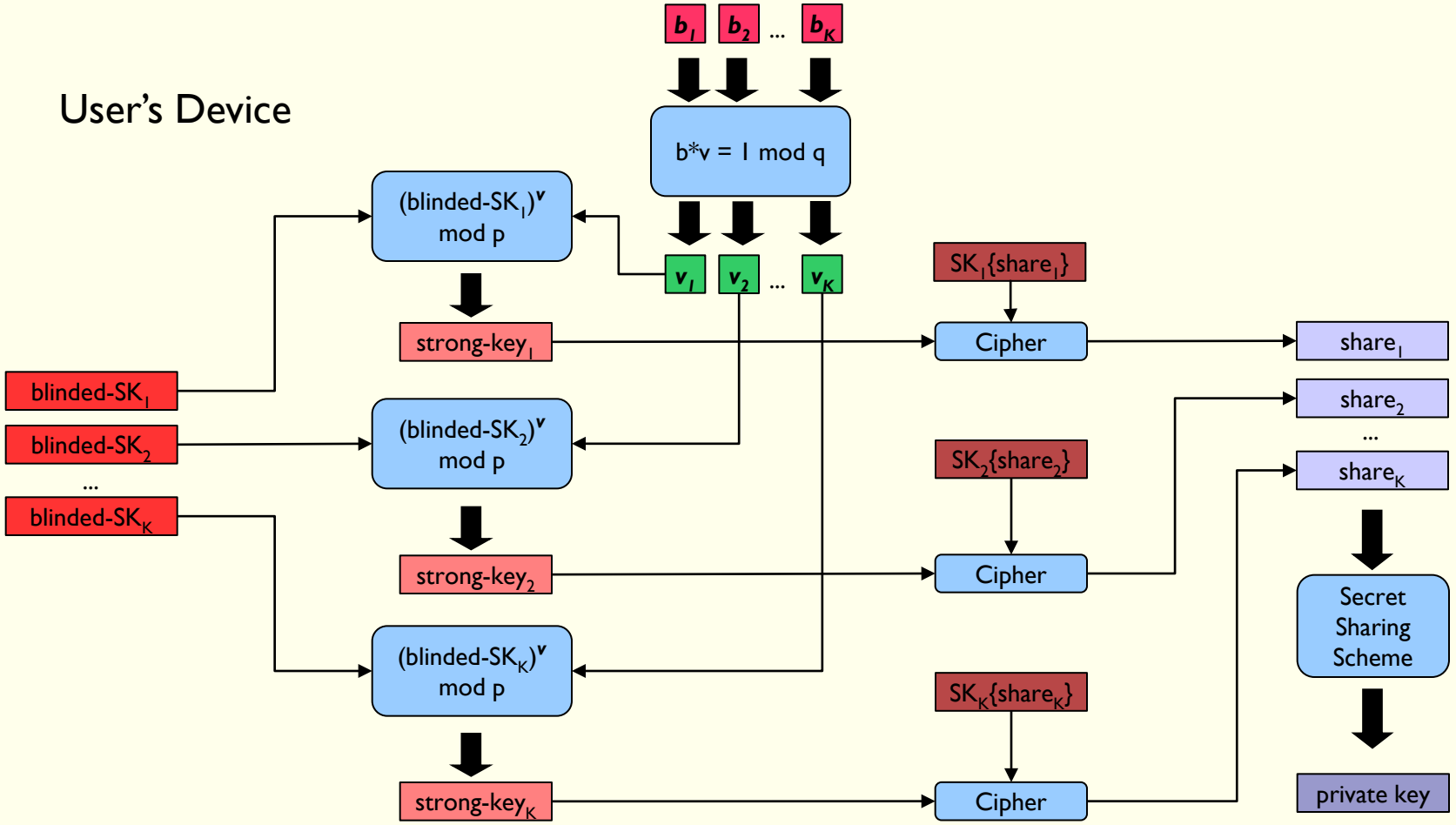




Distributed Key Retrieval (1 of 2)



Distributed Key Retrieval (2 of 2)



Why the math works

$p = 2q + 1$	P and Q - two large primes defined in the system
$x = f(password)$	Represent the f(password) with the number x
$strongkey = x^{2e} \pmod p$	Strong key is password to the power 2e
$((x^{2b})^e)^v \equiv x^{2e} \pmod p$	This is what will be proved…
$bv \equiv 1 \pmod q \Rightarrow bv = nq + 1$	Implies (bv)/q = n remainder 1, for some integer *n*.
$((x^{2b})^e)^v = x^{2bev} = x^{2e(nq+1)} = x^{2enq+2e}$	Substitute (bv) with (nq+1)
$x^{2enq+2e} = (x^{2q})^{en} \cdot x^{2e}$	Isolate the strong key
$(x^{p-1})^{en} \cdot x^{2e}$	Replace 2q with (p-1), since p = 2q+1
$1^{en} \cdot x^{2e} \pmod p$	By Fermat's little theorem: $a^{(p-1)} = 1 \pmod p$
$x^{2e} \pmod p = strongkey$	1 raised to any power is 1, this is the strong-key

Why the protocol is secure

- Blinded password is a function of two unknowns
  - Yields no information about password
- Clients do not learn the secret exponents
  - Requires interaction with a threshold number of authentication servers for each password attempt
- A correctly decrypted secret share cannot be distinguished from an incorrectly decrypted share
  - A threshold number of secret exponents are needed to conduct an offline dictionary attack

Security Analysis

- How long would it take to hack these systems:
  - Password stored across 5 different nodes:
    - (Like reliability of RAID 0 across 5 disks)
  - Private Key stored on a single node:
    - (Like reliability of data stored on a single disk)
  - Key Distributed over 5 different systems:
    - (Like reliability of erasure codes: can survive failures)

Applications

- Implementation of Java's Key Store
  - Seamless integration with many existing programs, works from any computer, nothing to brute force
- Web Browser plugin
  - Plugin turns user's password into a private key which can support client-side TLS authentication
  - Each authentication server run by different entity
- Distributed Key protocol on a Smart Card
  - If smart card is stolen, thief obtains nothing

Unsolved problems

- This protocol enhances security of service providers
- It does not save end users from themselves
  - Users must not share their password
  - The user's environment must be free of:
    - Malware, Key Loggers, and Shoulder Surfers
- Two-factor authentication still has merits for enhancing the security of the end-user's device

In Conclusion

- Distributed Keys offer value, especially for the cloud:
  - Secure, Available, Scalable, Easy, Flexible
- Through Distributed Keys, users can:
  - Have a single very good password for everything
  - Not worry that the hack of a random blog will put their data in the cloud or bank account at risk
- Through Distributed Keys, providers can avoid:
  - Telling customers they need to change their password everywhere because they were hacked

Questions and Answers

- Questions
  - And hopefully some answers!

- Example questions:
  - Isn't all the trouble on the end-user side?
  - Is PKI support a requirement to use this?
  - Is there a reference implementation?
  - Does this have applications beyond cloud storage?

References

- [1] Estimating password strength
  - NIST Special Publication 800-63, Version 1.0.2
- [2] How to Share a Secret
  - Adi Shamir, In Communications of the ACM 22 (11): 612–613, 1979.
- [3] Server-Assisted Generation of a Strong Secret from a Password
  - Warwick Ford and Burton S. Kaliski Jr. In Proc. IEEE 9th Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pages 176-180. IEEE Press, 2000.
- [4] Compromise of 10 million user passwords from Trapster:
  - http://blogs.computerworld.com/17690/over_10_million_passwords_possibly_compromised_at_trapster
- [5] Compromise of 2 million user passwords from SourceForge:
  - http://thenextweb.com/industry/2011/01/29/sourceforge-attacked-resets-2-million-account-passwords-to-protect-users/
- [6] Vulerability of Kerberos to offline dictionary attacks (RFC 1510, section 1.2):
  - http://www.ietf.org/rfc/rfc1510.txt
- [7] Compromise of 1.3 million user passwords from Gawker:
  - http://gadgetwise.blogs.nytimes.com/2010/12/13/gawker-passwords-hacked-what-you-should-do/