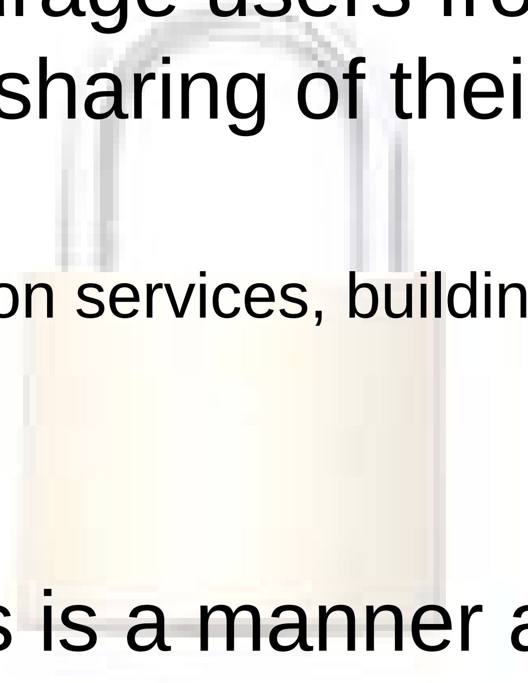
# Theft Protected Proprietary Certificates

A. Boldyreva and M. Jakobsson, “Theft protected proprietary certificates,” in *Proc. 2002 ACM Workshop on Digital Rights Management (DRM 2002).* Available http://crypto.stanford.edu/DRM2002/tppcertif.pdf, March 2003.

Presentation by Pene Geard

# Main Idea

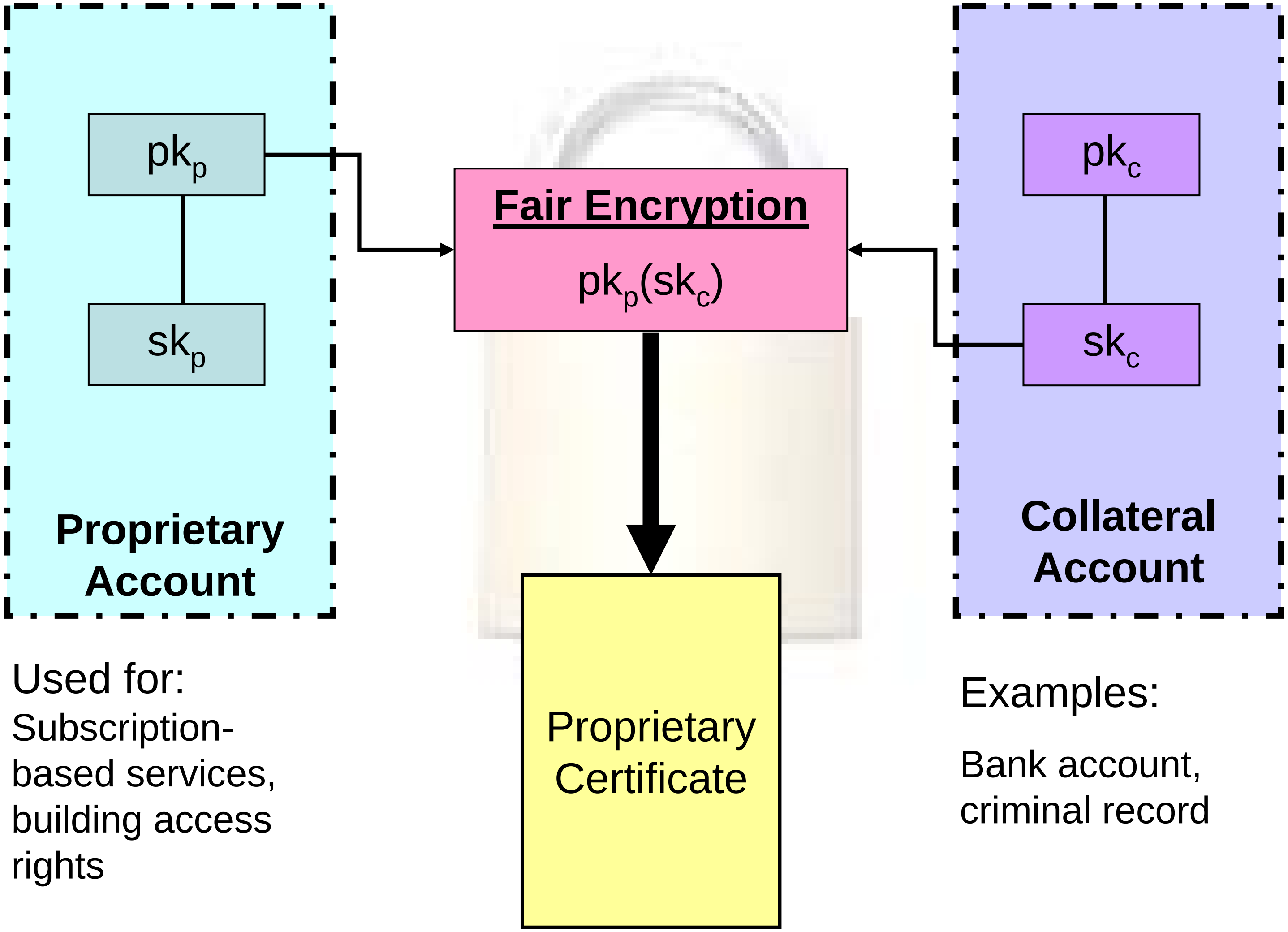


- How to discourage users from unauthorized sharing of their private/secret keys

  (eg. For subscription services, building access, etc…)

- How to do this is a manner acceptable to users

# Proprietary Certificates



Used for:

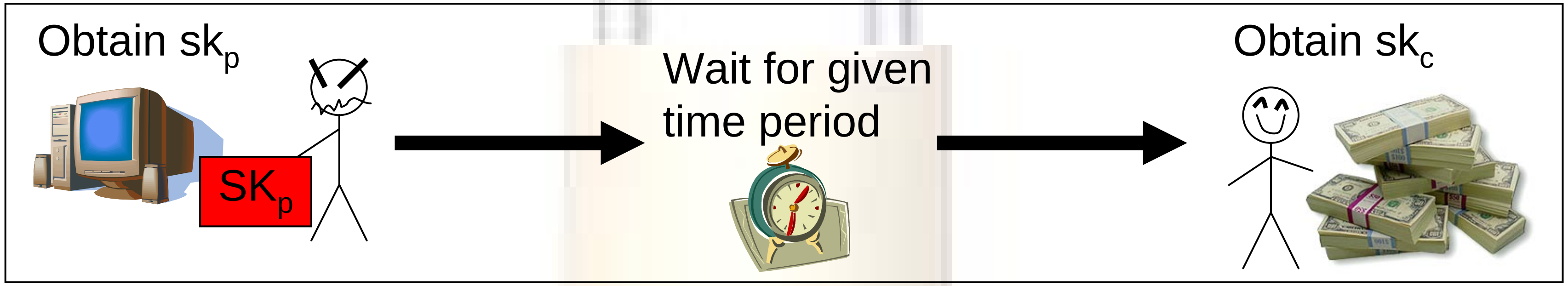
Subscription-based services, building access rights

Examples:

Bank account, criminal record

# Theft-Protection

- Introduces a time-delay in the decryption of the collateral secret key ($sk_c$)



Obtain $sk_p$

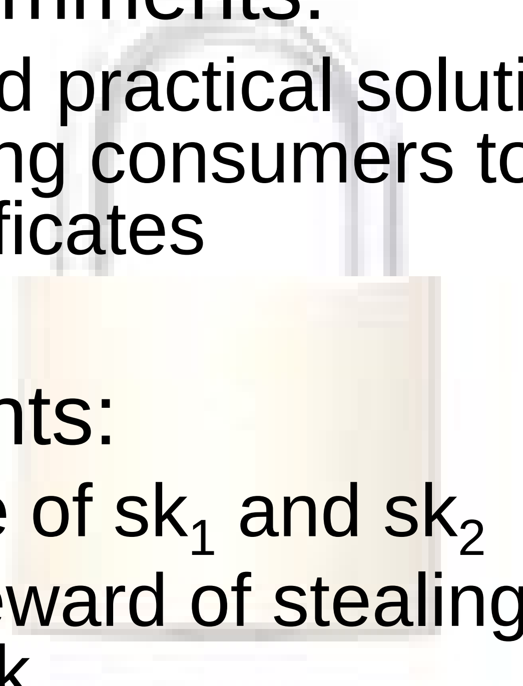
$SK_p$

Wait for given time period

Obtain $sk_c$

- Delay gives user time to detect theft and change keys

# Critique



## Appreciative Comments:

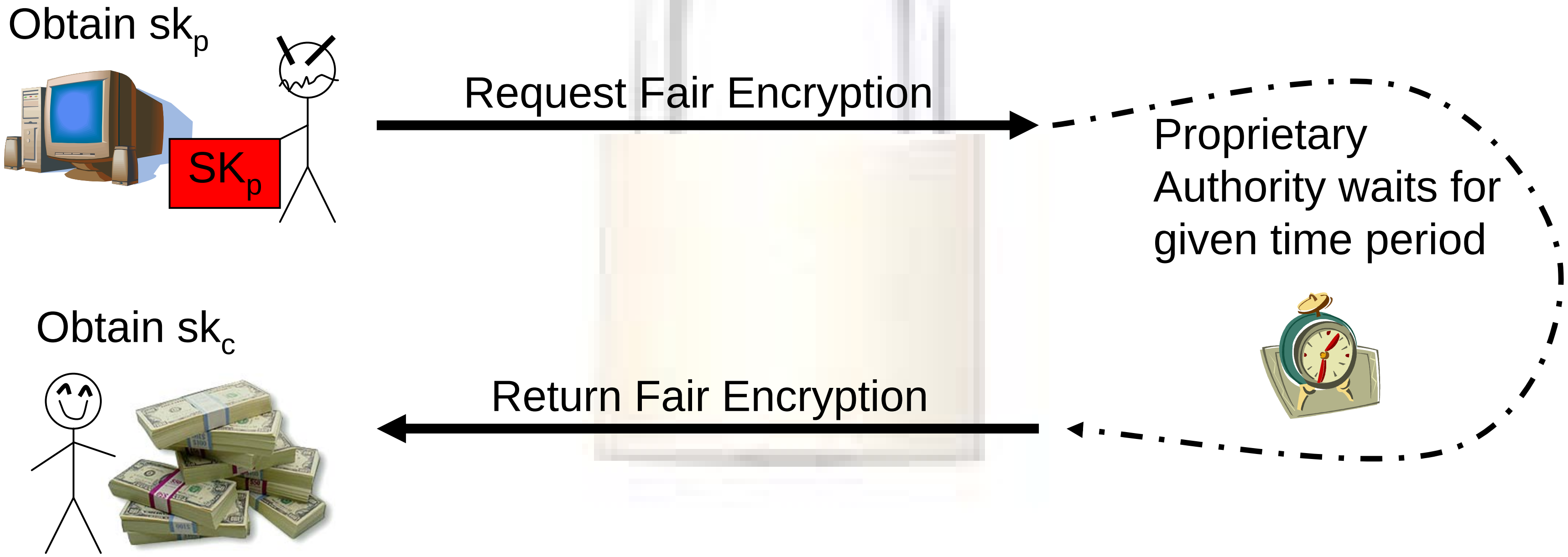
- Fairly simple and practical solution to the problem of getting consumers to accept/use proprietary certificates

## Critical Comments:

- Inconsistent use of $sk_1$ and $sk_2$
- Increases the reward of stealing $sk_p$ and reduces the security of $sk_c$
- Reduces effectiveness of proprietary certificates
- Issues with Theft-Detection/Notification

# Real Time Delay

- Introduces a time-delay in the decryption of the collateral secret key ($sk_c$)



Obtain $sk_p$

$SK_p$

Request Fair Encryption

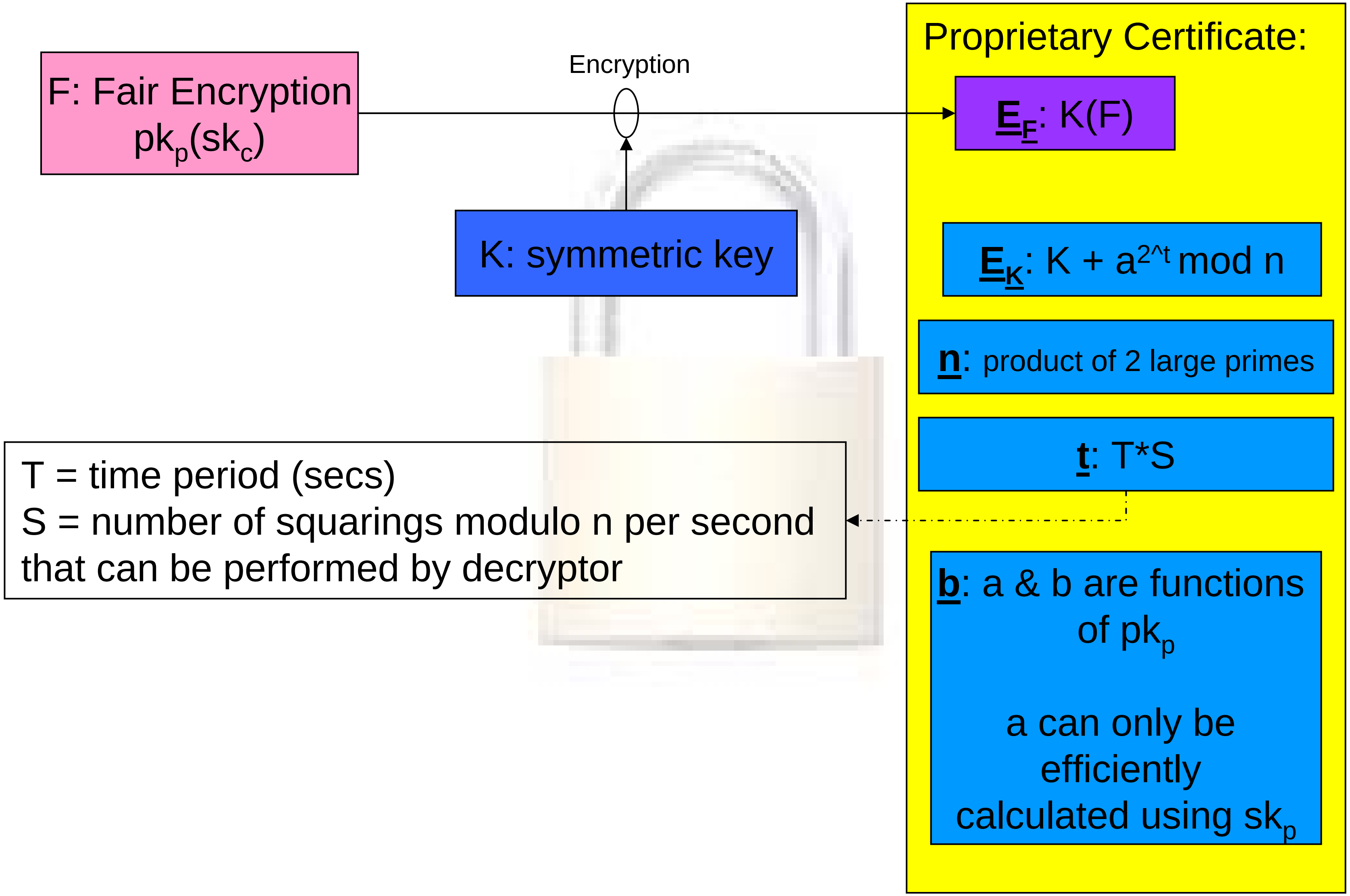
Proprietary Authority waits for given time period

Obtain $sk_c$

Return Fair Encryption

•Fair Encryption must be communicated securely or others might be able to avoid the time delay

# CPU Time Delay



F: Fair Encryption
$pk_p(sk_c)$

Encryption

K: symmetric key

T = time period (secs)
S = number of squarings modulo n per second that can be performed by decryptor

Proprietary Certificate:

**E_F**: K(F)

**E_K**: $K + a^{2^{\wedge}t} \bmod n$

**n**: product of 2 large primes

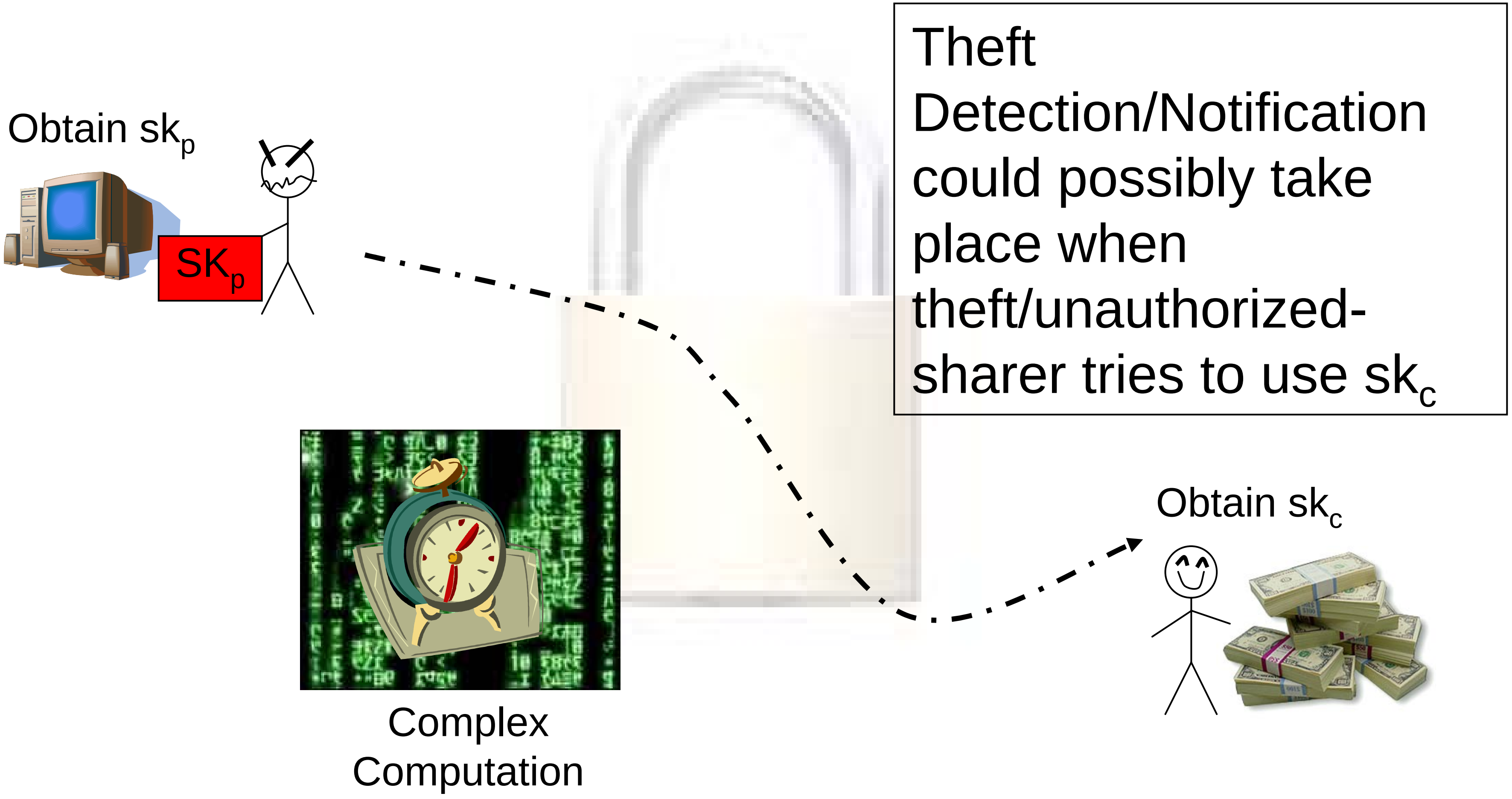
**t**: T*S

**b**: a & b are functions of $pk_p$

a can only be efficiently calculated using $sk_p$

# CPU Time Delay

- Time Delay due to computation time



Obtain $sk_p$

$SK_p$

Theft Detection/Notification could possibly take place when theft/unauthorized-sharer tries to use $sk_c$

Obtain $sk_c$

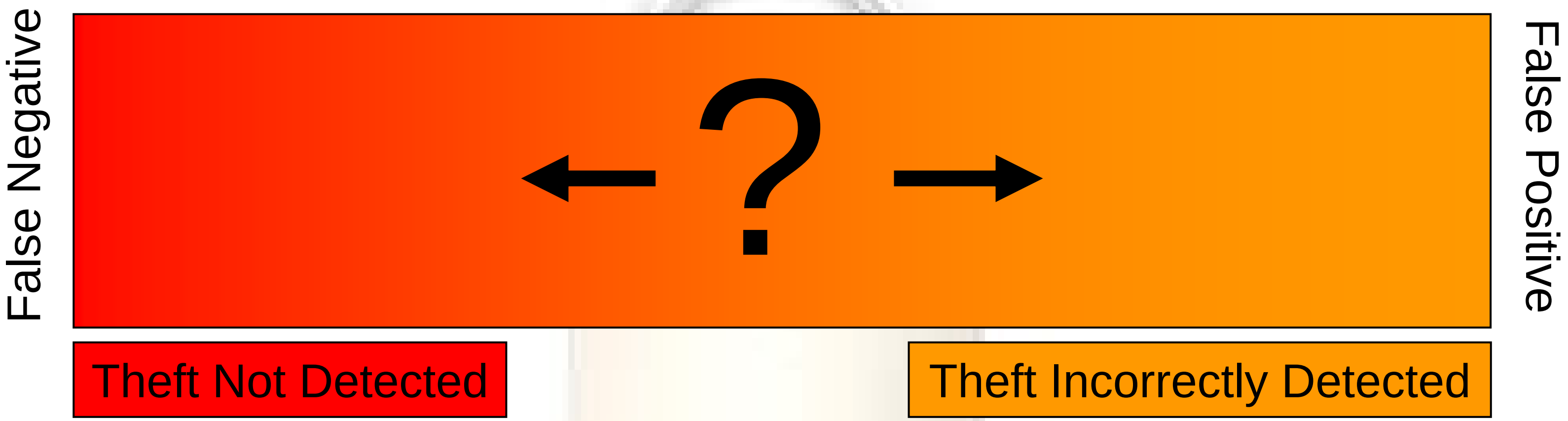
Complex Computation

# Theft-Detection

| | |
|---|---|
| True Negative<br>Unauthorized Sharing:<br>leads to loss of collateral key | True Positive<br>Theft Correctly Detected:<br>Collateral Key kept secret because of no unauthorized sharing |
| False Negative<br>Theft Not Detected:<br>User loses collateral key unfairly | False Positive<br>Theft Incorrectly Detected:<br>User gets away with unauthorized sharing by claiming theft |

# Discussion

What kind of balance is required?



False Negative

?

False Positive

Theft Not Detected

Theft Incorrectly Detected

Unfair loss of collateral key
could have severe effects
for the user

loss of profits/service
quality due to
unauthorized sharing