Project 4 - Lottery Security Design

Bernard Dickens III, Yan Liu, Jaime Arana-Rochel

Terms

1. State's server(s) [**SS**]: the lottery server(s) owned and operated by the state.
2. Lottery database [**LD**]: the database that exists on the state's lottery server(s); holds all the lottery's non-transient data.
3. Communication protocol [**CP**]: the method the lottery ticket terminal uses to communicate with the state's lottery server(s)/lottery database. Explained below.

Premises, Givens, and Assumptions

1. The SS employs (D)DoS mitigation via cheap load-balancing CDN.
2. The SS and LD are physically and digitally secured by the state.
3. All hashes are computed using SHA3.
4. SHA3 is not susceptible to length extension attacks, ergo less of a need for HMAC in this instance.
5. The clerks operating the lottery ticket terminals are not trustworthy.
6. The communication medium between SS and any lottery ticket terminal is not trustworthy.
7. Lottery ticket terminals are properly programmed to refuse dispensation of tickets during illegal hours (11:00 PM Saturday through 8:00 AM Monday).
8. A customer having to wait an average of several seconds for a ticket to print is presumed to be acceptable.
9. Lottery ticket terminals are tamper-proof/resistant; compromising the integrity of the terminal will break it. Further, each terminal has an accurate, tamperproof clock inside it.

Lottery Ticket Terminals

1. Each terminal has its own public and private key-pair baked into its hardware. The terminal hardware is tamper-resistant and will render the key-pair unrecoverable if the integrity of the terminal is compromised.
    a. **The terminal's ID is its <u>public key</u>.**
    b. If the terminal is stolen, the typical adversary gains nothing as the hardware is tamper-resistant. Further, even if the adversary did somehow obtain the terminal's private key and was able to forge valid ticket purchases to the LD, lottery officials can retroactively invalidate tickets in the LD that came from a specific terminal ID upon noticing the fraudulent activity or fraudulent attempt at cash redemption.
    c. If the physical integrity of a terminal is compromised, it will have to be completely replaced.
    d. The hardware protecting the key-pair can never be asked by the operating system or other hardware components to reveal its private key. The hardware API protecting the keys only exposes five operations: *publicKeyEncrypt, publicKeyDecrypt, getPublicKey, privateKeyEncrypt, privateKeyDecrypt*.
2. Each terminal has an internal USB port protected by a special physical lock. Accessing this port does not constitute a compromise of the terminal and will not invalidate the baked-in key-pair.
    a. Each terminal has a cheap low-capacity USB attached to it that houses the SS public key.
        i. This allows the state to maintain easy revocation rights for its key-pair in the case that the state's private key is leaked. This way, the entire terminal doesn't have to be replaced; instead, just the USB key gets swapped out.
    b. The terminal features no other physically accessible ports that can be interfaced with by potential adversaries.

    c. The operating system of the terminal will treat the USB as a read-only device and will never attempt to execute or otherwise interface with the data on the device. If the data on the USB key is not determined to be a valid public key through basic syntax checking or the USB is missing, the terminal will refuse to boot.
3. Each terminal communicates over WiFi in order to connect with the SS and the LD.
4. Each terminal immediately drops any incoming packets that are not a component of the CP. The lottery ticket terminals accept zero commands from any remote party, including the SS.
    a. Further (D)DoS and side-channel mitigation, this time from the client-side.
5. Each terminal is **not** required to keep track of the number of tickets sold. The SS does this using the LD.

State's Lottery Database (LD)

1. The database flags expired tickets (older than a week after that ticket's Sunday drawing) as expired but does not delete them for logistical and historical (financial audit) purposes. Expired tickets cannot be redeemed for the cash prize.
2. The database contains a lookup table detailing the 1-to-1 relationships between the hashed public keys of the various lottery ticket terminals and their actual public keys (as well as other information such as location, total ticket sales, etc) for the purposes of speedy lookup.
   table *terminals*: \$SHA3$_{100,000\text{-round}}$(pubK$_t$) -> (pubK$_t$, location$_t$, weekly_sales$_t$, *etc.*) for all terminals *t*
3. The database contains a table housing all of the lottery ticket terminal transactions in the state. For columns, it features: the hashed lucky_number || timestamp || name || birthday || cust_id || nonce, the nonce, the lucky_number, name, birthday, cust_id, the timestamp, and anything else the state needs.
   table *tickets*: \$SHA3$_{100,000\text{-round}}$(lucky_number || timestamp || name || birthday || cust_id || nonce) -> (nonce, lucky_number, name, birthday, cust_id, timestamp, *etc.*)
    a. cust_id is the customer's driver's license or state ID, kept for the purposes of verifying the identity of the customer
    b. The same customer (cust_id) cannot have two tickets with the same timestamp in the table, which shouldn't occur anyway

Communication Protocol (CP)

1. The SS never attempts to initiate communications with any of the lottery ticket terminals.
2. The lottery ticket terminal is **not open** to receiving commands or instructions of any kind over the internet, including from the SS.
3. The lottery ticket terminals communicate with the SS using **Enveloped Public Key Encryption** (EPKE). This ensures both non-repudiation (messages coming from each terminal are unique to that terminal) and confidentiality (only the SS can decrypt the communications) as guaranteed attributes. It works as follows:
    a. The lottery ticket terminal constructs "plaintext" message *m* which is composed of a nonce and 100,000-round* hash of the customer's chosen number, purchase timestamp, name, birthday, cust_id, and a randomly generated nonce.
       m = SHA3$_{100,000\text{-round}}$(lucky_number || timestamp || name || birthday || cust_id || nonce) || lucky_number || timestamp || name || birthday || cust_id || nonce
        i. Large round number is used at this point in order to reduce the efficacy of pre-computed/offline attacks and any potential brute-force attacks by counterfeiters or others down the road.
    b. The lottery ticket terminal encrypts the message *m* with its baked-in private key.
       privKE$_{\text{terminal}}$(*m*) = *c*

c. The lottery ticket terminal encrypts *c* with the SS public key obtained from the USB, first appending *c* with its hashed public key. PK was subjected to a 100,000-round***** hashing. $\text{pubKE}_{SS}(c \,||\, \text{SHA3}_{100,000\text{-round}}(\text{pubK})) = c'$
d. The lottery ticket terminal sends the SS *c'*, querying DNS for a route to the SS/CDN.
    i. We are not worried about MitM or DNS poisoning. Any adversary won't be able to tamper with or decrypt *c'* without failure anyway.
e. SS receives *c'*.
f. SS decrypts *c'* with its own private key.
$\text{privKD}_{SS}(c') = c \,||\, \text{SHA3}_{100,000\text{-round}}(\text{pubK})$
g. SS matches the hashed pubK of the terminal to a row in the *terminals* lookup table in the LD. No match causes an exception and halts the protocol.
h. Upon successful match, the SS decrypts *c* with the terminal's public key.
$\text{pubKD}_{\text{terminal}}(c) = m$
i. SS processes original message $m = \text{SHA3}_{100,000\text{-round}}(\text{lucky\_number} \,||\, \text{timestamp} \,||\, \text{name} \,||\, \text{birthday} \,||\, \text{cust\_id} \,||\, \text{nonce}) \,||\, \text{lucky\_number} \,||\, \text{timestamp} \,||\, \text{name} \,||\, \text{birthday} \,||\, \text{cust\_id} \,||\, \text{nonce}$
    i. SS stores the SHA3 hash, lucky_number, timestamp, name, birthday, cust_id, and nonce in LD
j. SS responds with a simple true/false indicating success or failure respectively. The response is also communicated to the lottery server using the inverse of the above protocol (EPKE). No response after a timeout period will be interpreted by the lottery ticket terminal as a failure (false).

Purchase Protocol

1. Customer wants ticket; Clerk accepts payment of $1 from customer
2. The customer then enters into the terminal:
    a. Their chosen lucky number or has the terminal choose for them.
    b. Their name as it appears on their driver's license or state ID.
    c. Birthday (must match format: MM/DD/YYYY) as it appears on their driver's license or state ID.
    d. Their driver's license or state ID number (referred to as **cust_id**) as it appears on their driver's license or state ID. Care is taken by the terminal to ensure that the number is valid and sanitized of invalid input before further processing.
3. The clerk presses a button on the opposite side of the terminal that signals to the terminal to dispense a ticket to the customer. The terminal then asks the user to verify what they entered and warns them that if any of the information is inaccurate (does not appear **exactly** as it does on their government identification card), they will not be able to redeem their ticket. It also performs heavy validation on the input from the user and will report error if the user entered something invalid.
4. If the user's input is valid, the lottery ticket terminal engages in the communication protocol (CP) described above.
5. **If the lottery ticket terminal interprets a fail response from the SS, the protocol does not proceed and the terminal prints a receipt stating that the ticket purchase failed.**
6. If the lottery ticket terminal interprets a success response from the SS, the protocol proceeds.
7. The lottery ticket terminal prints a lottery ticket. The ticket has a QR/barcode representing the original SHA3 100,000-round* hash of the ticket data, the SHA3 100,000-round* hash of the terminal ID (fingerprint), the chosen lucky number, and the timestamp. The two aforementioned hashes were already computed and cached during the CP and so cost no extra cycles to generate.

8. The lottery ticket terminal prints two copies of a receipt.
    a. The two receipts have the price of the ticket, the customer's information, as well as the purchase timestamp, terminal ID, and full hex-form of the QR/barcode that was printed on the original lottery ticket.
        i. The receipt **CANNOT** be substituted in place of the actual lottery ticket for verifying purchase to and/or collecting winnings from the state!
        ii. The receipt does NOT have the nonce listed on it.
    b. One copy of the receipt must be signed by the customer and given to the clerk, who keeps the ticket for the store's records. The other copy is given to the customer for his or her records.
        i. This provides protection in the case that the state is forced to invalidate all the tickets that were created by a specific terminal or terminals in a worst-case-scenario situation. In order to tell forgeries from actual tickets, the second factor of the verification process would be realized by going to the affected store or stores and matching receipt hashes (QR/barcodes) to ticket hashes (QR/barcodes). Tickets that were forged, even if they looked valid, would not have purchase receipts kept by the store. To defend against untrustworthy clerks, the state could go so far as to check the store's financial records and confirm parity with the store's receipts to ensure they were actually purchased.
    c. The customer is told to keep the receipt on their person and not to lose it or have it stolen. This is merely to protect the customer's information (such as their driver's license number) that is on the receipt. Losing the receipt does not compromise the lottery system but may prevent the user from redeeming their winnings in the case that something fraudulent or anomalous occurs.

*****The 100,000 round number can be adjusted at the factory depending on the computing power of the potentially-uber-cheap hardware. 10,000 rounds is the absolute lower bound. **A customer having to wait several seconds for a ticket to print is presumed to be acceptable.**

Weekly Drawing

After the weekly drawing occurs, the winning customer(s) have a week to collect their winnings as per the specification. In order to qualify as a winner, the following criteria must be met:

1. The QR/barcode on the customer's ticket must exist in the LD
2. The hashed public key of the terminal ID on the customer's ticket must map to a valid terminal public key in the LD
3. The customer must present the identification they used when purchasing the ticket. The customer's identifying information (name, birthday, cust_id), the timestamp and lucky number listed on the ticket, and the nonce in the LD are hashed together and compared both to the QR/barcode on the ticket *and* the hash in the LD. All three must match.
    a. The nonce is looked up in the LD using the the cust_id and timestamp.
    b. Hash is computed in the same order and manner that *m* was constructed during the CP: $\text{SHA3}_{100{,}000\text{-round}}$(lucky_number || timestamp || name || birthday || cust_id || nonce)
4. If the three-way hash match confirms the ticket as authentic, the LD is updated to reflect that the ticket has been redeemed. **This implies the winner cannot redeem the same hash (ticket) multiple times.**

Collecting Funds from Clerks

The state can audit the ticket sales records in the LD to ensure they're receiving their $0.90 per ticket sold per week from the stores housing each terminal. If a clerk or store has not given the government the amount of money calculated via number of ticket sales in the LD for the terminal in that store, the state can then take legal action.

This check will be performed weekly.

Threat Model
Threats outside of this model: *compromised or corrupt government officials, insecure government database/server, DoS attack against the state's servers/CDN, identity theft or identity forgery.*

Risk of a threat occurring: High (H), Medium (M), Low (L)

Risk	Threat	Reward	Mitigation
H	Insecure medium between SS and LD	MitM, eavesdropping, etc.	EPKE and distribution of PK via USB; MitM impossible.
M	Corrupt USB supplied to terminal	Manipulating terminal output and destination	Terminal confirms format of USB data before utilization.
L	USB with attacker's PK supplied to terminal	Manipulating terminal output and destination	SS will discard requests it cannot understand without question.
M	DNS poisoning redirects legitimate ticket sale data to attacker's server	Manipulating terminal output's destination	EPKE prevents attacker from decrypting data.
H	Terminal receives odd packets/commands over the network	Exploit vulnerabilities (network-facing) in terminal OS	Light-weight firewall drops "bad" packets.
H	(D)DoS attack against the terminal	Deny operation of terminals service	Light-weight firewall drops "bad" packets; **still potentially vulnerable against an extensive and powerful distributed attack.**
L	USB containing malware supplied to terminal	Manipulating terminal output and destination	Terminal confirms format of USB data before utilization.
L	Corrupt hardware component attempts to steal private key from internal terminal crypto API	Steal terminal private key and forge signed tickets	Hardware crypto component never exposes private key via API.
L	State's private key is leaked	Decrypt communications between SS and terminals	State maintains easy revocation rights of its key-pair.

M	A terminal’s private key is leaked/terminal is stolen	Forge signed tickets	State can invalidate all ticket purchases from a particular terminal in the LD.
H	Invalid input/injection attack against LD	Manipulate LD	Terminals sanitize input.
H	Customer receipt is stolen	Aid in forging tickets	Info on receipt is not enough to recreate ticket.
H	Customer lottery ticket is stolen	Attempt to redeem cash prize illegally	Tickets require government identity verification to be redeemed for cash prize.
M	Customer attempts to redeem their ticket more than once	Attempt to redeem cash prize illegally	LD invalidates redeemed tickets atomically; any further attempts at redemption are rejected.
M	Customer duplicates their own ticket	Attempt to redeem cash prize illegally	LD invalidates redeemed tickets atomically; any further attempts at redemption are rejected.
H	Adversary photocopies a winning ticket	Attempt to redeem cash prize illegally	Tickets require government identity verification to be redeemed for cash prize.
H	Customer alters a purchased ticket	Attempt to redeem cash prize illegally	Hash on ticket is matched with the hash in the database and a hash of the data on the ticket (and the nonce) plus the customer’s identity. All three must match. This is referred to as the **three-way hash match**.
M	Adversary forges a ticket and hash of a customer who already purchased a ticket	Attempt to redeem cash prize illegally	Adversary would need the nonce used in the original creation of the customer’s ticket, which he does not have.
H	Adversary forges a ticket and hash of a non-existent customer (ticket not in LD)	Attempt to redeem cash prize illegally	Adversary would need to inject a record of his forged ticket into the database, which he cannot do.
M	Untrustworthy clerk misreports ticket sales	Clerk attempts to make an extra buck on the side	LD keeps a record of tickets sold. The government can then ensure that they receive payment from the store that matches the number of tickets sold.

Criteria and Design Justification

It should be virtually impossible for anyone to create a bogus "winning" ticket (for example, by forging the ticket rather than buying it from a legitimate terminal, or by buying a legitimate ticket and changing its numbers or its timestamp) and cash in that ticket successfully.

It is virtually impossible to forge a ticket in the above system. In order for a ticket to be considered valid, a record of it must exist in the state's lottery database as described above. The adversary cannot inject his forged data into the database and so cannot forge a ticket from scratch. To forge the ticket of a customer already in the database (using their receipt), the adversary would need the nonce that was used to create that customer's hash, which they cannot generate. If the adversary were to instead buy a ticket and then alter it manually, he would still fail the three-way hash match when he walks in to collect the cash prize. If an adversary were to make a photocopy of the winning ticket, the adversary would have to provide proof-of-identity upon collecting the winnings which he shouldn't have. If a legitimate winner were to duplicate their winning ticket, the hash matching on the database side would result in a collision and one of the tickets would be recognized as a forgery.

If one of the terminals is stolen and/or hacked, circumventing its tamper-resistance and revealing its private key, or suspicious activity was detected from one or more terminals, it would then fall to the state to revoke the validity of all tickets issued from the compromised terminal(s) in their database. If, on the other hand, the state were to be hacked and their private key stolen, they can issue a revocation and generate a new key-pair, distributing their new public key via USB to the terminals via courier. From there, they can manually audit every store, matching receipts to financial records with the goal of determining which tickets are valid, though this is definitely an expensive worst-case scenario.

Regardless, it is virtually impossible to forge a ticket within the given threat model.

It should be difficult for a merchant to sell a ticket without eventually turning over ninety cents for that ticket to the lottery agency.

The state can audit the ticket sales records in the database to ensure they're receiving their $0.90 per ticket sold per week from the stores housing each terminal. If a clerk or store has not given the government the amount of money calculated via number of ticket sales in the database for the terminal in that store, the state can then take legal action.

This check will be performed weekly when receiving payment from the clerks/stores, as per the specification.

The cost of your design should be as low as possible.

Cost estimation

The cost estimation for one terminal per week is as follows:

1. We need a basic terminal does not contain any other functions, $50.
2. Add wireless link to each terminal, and make it possible to communicate and exchange data with the centralized server (the server on lottery center), $10.
3. Basic USB storage for the state's public key, $1.
4. Basic computation units (used for SHA3 hash computation), $1.
   We can reference the low-end cpu costs, which is enough for doing hash computations, for

example, https://www.cpubenchmark.net/cpu.php?cpu=Intel+Xeon+2.40GHz&id=1316. Since the computation units are added to the terminal it has no need to remove or update it, so the estimation of $1 per week is a fair estimation.

To sum up: $50+$10+$1+$1 = $62 for one terminal per week.

Additional one-time cost:

Supposing 10,000 lottery stores around the nation, hiring 50 couriers. Their job is to act as physical vectors for the key exchange (providing the terminals with the USBs containing the state's public key). The couriers only need to be hired for one week because this "key exchange" need happen only once, initially.

To sum up: $2000x50 = $10,000, which is the one-time cost for hiring these couriers.

This cost is justified, as the state now maintains easy revocation rights for their key-pair in a worst-case scenario. If the public key of the state's server were instead baked into the terminal at factory, in the case that the state's private key leaks, the state will have to switch out and completely replace *every single terminal in the state*, which will cost a lot of time, a lot of effort, and a lot of money.