Broadcast Encryption Based on the Naor–Reingold Pseudorandom Function[⋆]

Talley Amir[1]

Yale University, New Haven CT 06511, USA
talley.amir@yale.edu

Abstract. Broadcast encryption is a cryptographic tool that is used to send a message to a group of recipients. This paper provides a construction that realizes this functionality by leveraging the Naor-Reingold pseudorandom function. The security of the proposed scheme relies on the existence of unforgeable signature schemes, the existence of semantically-secure private-key encryption schemes, and the hardness of the discrete logarithm problem.

Keywords: Broadcast encryption · Secret sharing · Pseudorandom function.

1 Introduction

Broadcast encryption was first proposed by Fiat and Naor in 1993 [3]. This type of encryption scheme is used to securely communicate a single message to a target set of authorized users. Such functionality can be achieved trivially by using a public key encryption scheme to encrypt the same message to each of the individual users in the target set using their public keys. However, broadcast encryption schemes explore alternate mechanisms for achieving this same goal that improve on both the time and space complexity of the trivial solution. Broadcast encryption schemes can allow a broadcaster to publish a single ciphertext that can be decrypted by all and only the users in the target set.

The security of such schemes is defined by a notion called collusion resistance, which dictates that any subset of users who are not in the target set cannot pool their knowledge to recover the broadcaster's message. This security property is useful in scenarios where a single entity aims to communicate large private messages to groups of authorized parties. For example, broadcast encryption can be used to securely deliver a television service to paying subscribers, or to achieve access control in a network with unsecured channels.

1.1 Previous and Related Works

The most obvious of the trivial solutions, mentioned previously, relies on public key cryptography. This solution allows a broadcaster to communicate a message

[⋆] Supported by Yale University

to a target set of users by encrypting the same message individually to each of the users in the set under their public keys. Then, each individual can privately decrypt the ciphertext they received from the broadcaster using their private key. The primary drawback of this solution, as noted by Fiat and Naor [3] is its time and space inefficiency. This solution requires each user to have their own public and private key pair and to register the public key with the broadcaster via some certificate-issuing mechanism. After this elaborate setup procedure, each broadcast will scale with the size of the target set which may be very large, in the worst case being equal to the set of all users in the system.

Another obvious, but clearly unsuitable, solution is to use a reliable consensus protocol to broadcast messages, such as [2]. This would reduce the size of the message needed to be sent by the broadcaster, and ensure correctness of the message received by all parties in the target set. However, it would not guarantee privacy, and therefore, any user outside the target set can eavesdrop on the communication and recover the broadcasted message. Thus, such a scheme does not satisfy collusion resistance and therefore is not secure.

Alternatively, we can define a system in which a separate symmetric key is defined for all possible subsets of the users in the universe, as detailed in [3]. Then, in order to encrypt a message to a particular subset, the broadcaster can use that subset's corresponding key and then post the resulting ciphertext publicly. In this scheme, every user must store locally all keys it would need to decrypt a message targeted at any subset of the universe of which that user is a member. This means that if there are N users in the universe, each user needs to store 2^{N-1} keys. In addition, the broadcaster needs to store 2^N keys in order to encrypt to each possible subset. Not only is this extremely space-inefficient, but this system does not scale well. If any user wants to join the system, the broadcaster and every individual user needs to update their key set.

Another option is to use the same idea of attributing a unique key to each subset in the universe, but instead of precomputing and storing all of these keys at setup time, somehow securely transmit these keys as they are needed. For example, Naor, Naor, and Lotspiech (NNL) trees [1], the Goldreich-Goldwasser-Micali (GGM) pseudorandom generator, and secret sharing can all be used to derive and distribute encryption keys. When a broadcaster publishes a ciphertext, there is some mechanism in place for transmitting the key needed to decrypt the ciphertext to the target set of authorized users.

1.2 Our Contribution

In practice, secret sharing is frequently used to distribute encryption keys among a group of users. Secret sharing is a cryptographic primitive that is used to hide a secret unless some threshold number of users collaborate to recover the secret. In this paper, we will define a secret sharing scheme that is based on the Naor-Reingold pseudorandom function that can be used to achieve a resilient broadcast encryption scheme.

This scheme achieves maximal collusion-resistance; in particular, the broadcast message cannot be recovered by members outside the target set even if *all* such members are colluding. Moreover, the scheme is resilient to an actively malicious adversary that aims to force the target set to recover a message that differs from the originally broadcast message.

2 Security Notions

2.1 Definitions

A broadcast encryption scheme $\Pi_{\text{Broadcast}}$ is defined by a tuple of algorithms $(\mathsf{Setup}, \mathsf{Broadcast}, \mathsf{Recover})$ that operate over a message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and set of users $\mathcal{U}$. Our goal is to construct such a scheme with the following properties:

- **Correctness**: If a trusted party runs the Setup algorithm and obtains a ciphertext $C \leftarrow \mathsf{Broadcast}(M, T)$ for $M \in \mathcal{M}$, $C \in \mathcal{C}$, and $T \subseteq \mathcal{U}$, then $[M_1, ..., M_N] \leftarrow \mathsf{Recover}(C)$ produces $M_i = M \; \forall \, i \in T$.
- **Collusion-resistance**: Even if all users not in the target set T are colluding, if a trusted party runs Setup and computes $C \leftarrow \mathsf{Broadcast}(M, T)$, then $[M_1, ..., M_N] \leftarrow \mathsf{Recover}(C)$ produces $M_i = \perp \forall \, i \notin T$.
- **Resilience**: Even if all users not in the target set T are colluding, if a trusted party runs Setup and computes $C \leftarrow \mathsf{Broadcast}(M, T)$, then $[M_1, ..., M_N] \leftarrow \mathsf{Recover}(C)$ produces $M_i = M \; \forall \, i \in T$ with all but negligible probability.

2.2 Security Game

Let $\Pi_{\text{Broadcast}} = (\mathsf{Setup}, \mathsf{Broadcast}, \mathsf{Recover})$. Define the security game $\mathsf{Adv}^{\text{Broadcast}}_{\Pi,\mathcal{A}}$ as follows:

1. Let $N = |\mathcal{U}|$. A trusted party computes $(pk, \{sk_i\}_{i=1}^N) \leftarrow \mathsf{Setup}(1^\lambda, N)$, publishes the public parameter pk, and securely distributes the private keys $\{sk_i\}_{i=1}^N$ to each of the corresponding users $i \in [N]$.
2. The same trusted party picks a message M and a target set of users $T \subseteq [N]$ and computes and publishes $C \leftarrow \mathsf{Broadcast}(M, T)$.
3. A probabilistic polynomial-time adversary $\mathcal{A}$, who can view the public parameter pk and ciphertext C, corrupts any number of users in the set $[N] \backslash T$. The adversary can learn these users' secret keys and send messages to each other and to users in T.
4. All users in $[N]$ collectively participate in the $\mathsf{Recover}(C)$ algorithm. After performing the algorithm, each user $i \in [N]$ has a locally stored message M_i.
5. $\mathsf{Adv}^{\text{Broadcast}}_{\Pi,\mathcal{A}}$ outputs 1 if $\mathcal{A}$ can guess any bit of M_i for $i \in T$ with probability non-negligibly better than $\frac{1}{2}$ *or* if it causes any member i of the target set T to recover a message $M_i \neq M$. Otherwise, $\mathsf{Adv}^{\text{Broadcast}}_{\Pi,\mathcal{A}}$ outputs 0.

We call a broadcast encryption scheme Handsome-Dan-secure if there exists negligible function negl such that:

$$\Pr[\mathsf{Adv}^{\text{Broadcast}}_{\Pi,\mathcal{A}} = 1] < \mathsf{negl}(\lambda)$$

3 Proposed Solution

Let $\Pi_{\text{Sign}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be an unforgeable signature scheme, and let $\Pi_{\text{Enc}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a semantically-secure private-key encryption scheme. Define $\Pi_{\text{Broadcast}} = (\mathsf{Setup}, \mathsf{Broadcast}, \mathsf{Recover})$ as follows:

- $((\{vk_i\}_{i=1}^N, \mathbb{G}, q, g), \{sk_i\}_{i=1}^N) \leftarrow \mathsf{Setup}(1^\lambda, N)$:
  - Interpret the inputs as the security parameter 1^λ and the total number of users in the universe N. Users are enumerated from 1 to N.
  - Define a finite group $\mathbb{G}$ of order q with generator g.
  - For i from 1 to N, compute $(sk_i, vk_i) \leftarrow \mathsf{Gen}(1^\lambda)$.
  - Securely distribute each signing key sk_i to the corresponding user i.
  - Publish $pk = (\{vk_i\}_{i=1}^N, \mathbb{G}, q, g)$.
- $C \leftarrow \mathsf{Broadcast}(M, T)$:
  - Interpret the inputs as the message M to be broadcast to $T \subseteq [N]$.
  - Pick a random element $r \leftarrow \mathbb{Z}_q$ and compute g^r.
  - Let $v = [r, sk_1, sk_2, ..., sk_N] \in \mathbb{Z}^{N+1}$ and $u = [1, \mathbb{1}(1), \mathbb{1}(2), ..., \mathbb{1}(N)] \in \{0,1\}^{N+1}$, where $\mathbb{1}(i)$ is 1 if $i \in T$ and is 0 otherwise. Let F be the Naor-Reingold PRF [4]. Compute $R = F_v(u) = g^{r \prod_{i \in T} sk_i}$.
  - Use the bits of R as the random coins in the KeyGen algorithm to compute symmetric key $K \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  - Compute $c \leftarrow \mathsf{Enc}(K, M)$.
  - Output $C = (g^r, c, T)$.
- $M' \leftarrow \mathsf{Recover}(C)$:
  - Interpret the input as $C = (g^r, c, T)$. The recovery algorithm runs in three phases: First, the users in the target set individually contribute to the computation of a random seed. Second, the random seed is propagated to all users in the target set. Finally, each member of the target set uses the random seed to compute the decryption key and decrypt the broadcast message. Let $T(1), ..., T(t)$ denote the users in T sorted by index number (where $t = |T|$).
  - *Phase 1*: For i from 1 to t,
    - If $i = 1$, $T(1)$ computes $y = (g^r)^{sk_{T(1)}}$ and $z = \mathsf{Sign}_{sk_{T(1)}}(y)$. It then forwards (y, z) to $T(2)$.
    - If $i \neq 1$, $T(i)$ waits to receive a value X. It interprets $X = (y, z)$. If $\mathsf{Verify}(vk_{T(i-1)}, y, z) \neq 1$, it ignores the value and continues to listen for another message. Otherwise, it computes $y' = y^{sk_{T(i)}}$ and $z' = \mathsf{Sign}_{sk_{T(i)}}(y')$ and forwards (y', z') to $T(i+1)$ (or $T(1)$ if $i = t$).
  - *Phase 2*: For i from 1 to $t-1$,
    - If $i = 1$, $T(1)$ receives X which it interprets as $X = (y, z)$. If $\mathsf{Verify}(vk_{T(t)}, y, z) \neq 1$, it ignores the value and continues to listen for another message. Otherwise, it locally stores $R = y$, computes $z' = \mathsf{Sign}_{sk_{T(1)}}(y)$ and sends (y, z') to $T(2)$.
    - If $i \neq 1$, $T(i)$ waits to receive a value X. It interprets $X = (y, z)$. If $\mathsf{Verify}(vk_{T(i-1)}, y, z) \neq 1$, it ignores the value and continues to listen for another message. Otherwise, it locally stores $R = y$, computes $z' = \mathsf{Sign}_{sk_{T(i)}}(y)$ and forwards (y, z') to $T(i+1)$.

- *Phase 3*: For i from 1 to N,
  - If $i \in T$, i uses the bits of the locally stored value R as the random coins in computing $K \leftarrow \mathsf{KeyGen}(1^\lambda)$. It then computes and stores $M_i = \mathsf{Dec}(K, c)$.
  - If $i \notin T$, $M_i = \perp$.

4 Security Analysis

4.1 Proof of Correctness

Claim: If the users behave honestly, then this scheme satisfies correctness.

Proof. Let $((\{vk_i\}_{i=1}^N, \mathbb{G}, q, g), \{sk_i\}_{i=1}^N) \leftarrow \mathsf{Setup}(1^\lambda, N)$ be run by a trusted party and let $C \leftarrow \mathsf{Broadcast}(M, T)$ for $M \in \mathcal{M}$ and $T \subseteq [N]$. We will show that $[M_1, ..., M_N] \leftarrow \mathsf{Recover}(C)$ must be such that $M_i = M$ for all $i \in T$.

We assume all users in the target set are honest, therefore the Recover algorithm is performed as specified above: Each user waits until they receive a message from the previous user and no messages are dropped nor intercepted. Then as the protocol is defined, the users will pass values to each other within the target set in their natural ordering. Starting with the user of the lowest index i_1, the user computes $(g^r)^{sk_{i_1}}$ and passes this value to the next user who computes $((g^r)^{sk_{i_1}})^{sk_{i_2}}$, and so on. Thus the value that gets passed back to the first user in the target set (that is then propagated to the rest of the users) is $g^{r \cdot sk_{i_1} \cdot sk_{i_2} \cdot ... \cdot sk_{i_t}}$. Notice that this is exactly equal to the Naor-Reingold pseudorandom function with seed $[r, sk_{i_1}, ..., sk_{i_N}]$ evaluated on input a binary string whose first bit is 1 and whose ith bit represents whether user i is in the target set or not (for $i > 0$). Thus the key computed by the target set of users, which uses this randomness to compute the symmetric key, is exactly equal to the key the broadcaster uses to encrypt M, as needed in the private-key symmetric setting.

The correctness of $\Pi_{\text{Broadcast}}$ is thus reducible to the correctness of the Π_{Enc} encryption scheme, which we define to be a private-key symmetric encryption scheme that satisfies correctness.

4.2 Proof of Security

Claim: If Π_{Sign} is an unforgeable signature scheme and Π_{Enc} is a semantically-secure private-key encryption scheme, then $\Pi_{\text{Broadcast}}$ is Handsome-Dan-secure.

Proof. In order to prove this claim, we must show that if $\Pi_{\text{Broadcast}}$ is not Handsome-Dan-secure, then either Π_{Enc} is not a semantically secure private-key encryption scheme or Π_{Sign} is not an unforgeable signature scheme.

For the sake of contradiction, let $\mathcal{A}$ be a p.p.t. algorithm that breaks the Handsome-Dan-security of $\Pi_{\text{Broadcast}}$. Thus, either it is the case that $\mathcal{A}$ is able to guess with non-negligible advantage some bit of the message encrypted and published by the broadcaster, or it is the case that $\mathcal{A}$ can cause the target set to recover a message not equal to the message encrypted by the broadcaster.

Note that these cases are not necessarily mutually exclusive; nonetheless, let us consider each of these cases separately. We will show that in either case, one of our assumptions is contradicted.

Consider the case where $\mathcal{A}$ can cause the target set to recover a message not equal to the original message encrypted and posted by the broadcaster. As shown previously, if the target set performs the algorithm honestly, it *must* recover exactly the original message. Therefore, the only way for the adversary to influence the outcome is to inject a message into the algorithm by sending a message to one of the users that changes the key that the target set recovers. It can only do this by sending a message (y, z) such that $y = g^\gamma$ for some randomly sampled value γ, which would cause each subsequent user in the algorithm's protocol to raise their secret key to that value. The resulting value would then get propagated to every member of the target set, and cause all parties to decrypt using the wrong key (which would either result in an incorrect value, or $\perp$, depending on how we define Π_{Enc}).

Notice that each subsequent user would not be able to detect that the y values they receive are incorrect due to the hardness of the discrete logarithm problem. In particular, if a user can detect that the value they receive is not correctly computed, they must be able to determine that the exponent of the value they receive is not the product of the random value r (unknown to the user by the discrete logarithm hardness assumption) and the sk_i of the previous users in the protocol (also unknown to the user). Clearly this cannot be done, and so the y value that the user interprets is indistinguishable from random. However, in order for the user that receives the message from the adversary to forward the next value to the next user in the protocol, the user must first verify the accompanying signature z on y. Thus, if the adversary causes the target group to recover another message by influencing the key recovered in the Recover algorithm, it must have correctly sent a forged signature z on y that verifies under $vk_{T(i-1)}$ to party $T(i) \in T$. This would mean that the adversary has broken the unforgeability of Π_{Sign}, which we assume to be unforgeable, thereby resulting in a contradiction. Thus we know that $\mathcal{A}$ cannot cause the target set to uncover a different message.

Now let us consider the case where $\mathcal{A}$ is able to guess with non-negligible advantage any bit M^i of the message M encrypted and posted by the broadcaster. Since the pairwise channels between each of the users in the system are secure, $\mathcal{A}$ cannot learn the secret key recovered by the users in the target set for the particular message. However, it can use values computed for previous messages that may be remembered by the users it corrupts for this message. Namely, it may know $\{sk_i\}_{i \notin T}$, and it also may know intermediary computations from previous executions of Recover, such as (y, z), or a previously recovered key.

Note that previous keys are only valid for a specific target group and ephemeral key g^r. Thus, for a new broadcast with new target group and new ephemeral key, a previous key cannot be modified to compute a new key because the ephemeral key changes the resulting key. For example, say that the broadcaster encrypts a message to two users, Alice and Bob, and the ephemeral key is g^{r_1}. In another broadcast, the target group consists of only Bob, and the ephemeral key is g^{r_2}, where $r_1 \neq r_2$. Then Alice cannot obtain the key used to decrypt the message targeted only at Bob by simply "removing" her signing key from the exponent of the key computed previously for the message targeted at only her and Bob because there is a new quantity in the exponent, r_2. Alice cannot compute the key only directed at Bob without knowing r_1 *and* r_2, which would require her to solve the discrete logarithm problem. Additionally, the intermediary computations that $\mathcal{A}$ has access to reveal nothing about the final key resulting from the Recover protocol, nor the private signing key of each user, because of the discrete logarithm assumption. Thus, $\mathcal{A}$ can only determine the private key of the target group if it corrupts users within the target group, which it cannot.

As a consequence, the key cannot be recovered by $\mathcal{A}$ because $\mathcal{A}$ is not allowed to corrupt users in the target set. Thus if it can infer a bit of M from $C = (g^r, c, T)$, of which g^r and T are irrelevant to M, it must do so by inferring something about M from $c \leftarrow \mathsf{Enc}(K, M)$. This breaks the semantic security of Π_{Enc}, which we assume to be a semantically secure private-key encryption scheme. Thus we also know that $\mathcal{A}$ cannot guess any individual bit of M with non-negligible advantage.

In conclusion, $\mathcal{A}$ has negligible probability of either forcing the target set to recover a message not equal to the original message encrypted by the broadcaster or guessing correctly better than randomly any bit of the message. Therefore, there exists negligible function negl such that $\Pr[\mathsf{Adv}^{\text{Broadcast}}_{\Pi,\mathcal{A}} = 1] < \mathsf{negl}(\lambda)$, and so this scheme is Handsome-Dan-secure.

5 Use Cases and Properties

As proven above, this paper provides a construction for a secure, resilient broadcast encryption scheme. This construction provides several desirable qualities that have many useful applications in practice. This construction is resilient in that it precludes outside parties from interjecting false computations into the Recover algorithm by using digital signatures to validate messages communicated between users. It also guarantees that parties cannot obtain the key for a message unless they are in the target group *for that specific message* because of the ephemeral key which makes computing the private encryption key equivalent to solving the discrete logarithm problem.

Unlike various other constructions, this construction allows users to enter and leave the system very easily. The only modifications that need to be made in order to accommodate new users is to generate an additional sign-verify key pair for the new user, to give the broadcaster access to the signing key, and to post publicly the verification key.

The motivation for using this particular scheme is that it allows us to use this "trivial solution" of using a different key for each possible subset of users, but allows us to compute these keys "on the fly," thereby reducing the storage load on each individual user. This scheme is particularly advantageous in systems where the broadcaster has users frequently entering and leaving the system. Another advantage of this scheme is that it uses symmetric key encryption, which is very fast. Moreover, the key itself is fast to compute by the broadcaster because the Naor-Reingold pseudorandom function is fast to compute.

5.1 Opportunities for Improvement

Unfortunately, this scheme also has some very prominent drawbacks. The algorithm for computing the private signing key among the users in the target set is very slow, having to be computed in series and traverse the entire group twice. This means that the time to compute the key scales with the size of the target set. As a result, this scheme is better suited for systems wherein the the broadcaster frequently needs to target unique small subsets of the universe of users.

Another notable downside to this approach is that it requires secure channels between every pair of users. This can be achieved fairly straightforwardly by giving each party a public-private key pair and encrypting messages directed to each party using their public key. However, this would require additional key storage overhead, which we aim to minimize. In addition, this scheme does not account for what happens if messages are dropped or intercepted – the Recover algorithm would simply halt mid-computation, and fail to output a decryption.

5.2 Known Attacks

Finally, it is important to note that replay attacks are still possible under this scheme. This can be illustrated by a small example. Consider two broadcasts $C_1 = (g^{r_1}, c_1, \{1,2,3,4\})$ and $C_2 = (g^{r_2}, c_2, \{2,4\})$. An adversary can use the signature sent to 3 in the recover algorithm performed for C_1 and inject that into the recover algorithm for C_2. Notice that the adversary can corrupt 3 in the recover algorithm for C_2 because $3 \notin T_2$. Thus the signature sent to 3 for C_1, which was $(g^{r_1 sk_1 sk_2}, \mathsf{Sign}_{sk_2}(g^{r_1 sk_1 sk_2}))$, is a valid signature verifying under vk_2. When the adversary sends this signature to user 4 during the recover algorithm for C_2, 4 will correctly verify the signature under the previous party's (2's) verification key. However, this will then cause the target group to recover the random value $g^{r_1 sk_1 sk_2 sk_4}$, which is not the desired value $g^{r_2 sk_2 sk_4}$. As a consequence, the recovered key will be incorrect, and will result in decryption failure. Future works can try to patch this security vulnerability by including a timestamp in the signature verification algorithm.

6 Conclusion

All in all, this scheme achieves a secure mechanism for broadcasting messages to a target set of users that requires the broadcaster to perform minimal computation, engage in minimal communication, and achieve constant ciphertext size (with respect to the size of the target set). However, the computational and communication burden is then put on the target set. Systems that would benefit from this type of broadcast encryption scheme are ones where the broadcaster frequently needs to broadcast to different small groups of users. Future works should investigate whether the same scheme can be modified to achieve the same security with less communication between users over unsecured channels.

Acknowledgements I would like to thank Mariana Raykova for advising me in this project, as well as Samuel Judson and James Miller for providing suggestions for revision.

References

1. Bhattacherje, Sanjay. Sarkar, Palash. Tree Based Symmetric Key Broadcast Encryption. *Indian Statistical Institute.* (2015).
2. Bracha, Gabriel. Asynchronous Byzantine agreement protocols. Information and Computation, 75(2):130143, November 1987.
3. Fiat A., Naor M. (1994) Broadcast Encryption. In: Stinson D.R. (eds) Advances in Cryptology CRYPTO 93. CRYPTO 1993. Lecture Notes in Computer Science, vol 773. Springer, Berlin, Heidelberg.
4. Naor, Moni; Reingold, Omer (2004), "Number-theoretic constructions of efficient pseudo-random functions", Journal of the Association for Computing Machinery, 51: 231262.