A P2P Market Place Based on Aggregate Signatures

Dario Catalano[1], Giancarlo Ruffo[2], and Rossano Schifanella[2]

[1] École Normale Supérieure, Paris, France
[2] Dip. di Informatica - Università di Torino, Italy

Abstract. A peer-to-peer market place is likely to be based on some underlying micro-payment scheme where each user can act both as a customer and as a merchant. Such systems, even when designed for largely distributed domains, may be implemented according to hybrid topologies where trusted third intermediaries (e.g. the broker) are single points of failures. For this reason it is crucial that such central entities scale well w.r.t. the overall number of transactions. In this paper, we focus on PPay as a case study, to show how the broker would greatly benefit in terms of computational cost if aggregate signatures are adopted instead of RSA signatures.

1 Introduction

Incentives and micro-payments can be used to stimulate the users [1] and to avoid the free-riding phenomenon [2], and they are largely used in practice, e.g., BitTorrent [3], EMule [4] and Mojo-Nation[1]. In particular, a micro-payment scheme is an interesting alternative to a differential service incentives, especially when a market place is layered on top of a p2p system. Current peer-to-peer micro-payment schemes use an hybrid topology, because some central units (e.g., the broker, the certification authority) are needed. For example, PPay [5] is based on the idea of "transferable coins". Basically a tranferable coin allows a user to either cash it, by interacting with the broker, or to re-assign it to other peers. The second alternative has been introduced for fault tolerance reasons, because when millions of transactions occur during a short time period, the broker is likely to be responsible of many concurrent, computationally expensive, operations (such as digital signature verifications and generations). Moreover, the broker should be able to detect frauds (e.g. duplicate coins) and then it has to store all the information related to forged coins for future checkouts. For this reason coins should be kept on floating for a while, before the broker is asked to cash them. At the same time, digital coins should not (excessively) grow in size after each re-assignment.

The choice of a coin re-assignment strategy that is scalable in terms of the overall number of transactions is thus of crucial importance: as the broker is a single point of failure, the best (in terms of both space and time) assigment strategy should be used in a practical market place. For example, *FairPeers* [6], a p2p application that allows profit and file sharing, uses PPay coins extensively, and the entire system would break down if the broker is overwhelmed by an inefficient strategy.

[1] At the time of this writings the beta version of Mojo-Nation platform has been shut down by its creator Jim McCoy. He announced that another project will get the heritage.

G. Chen et al. (Eds.): ISPA Workshops 2005, LNCS 3759, pp. 54–63, 2005.

In this paper, we compare different coins management policies, introducing also a novel approach based on the idea of aggregate signatures [7]. PPay and aggregate signatures are briefly introduced respectively in Section 2 and 3. In Section 4 we outline the comparative model, and the results of our analysis are given in Section 5.

2 An Overview of PPay

PPay, proposed by Yang and Garcia-Molina [5], tries to minimize the interaction with the broker allowing direct transactions between peers. To this end, they suggested the idea of *floating and self-managed currency*. The coins can float from one peer to another, and the owner of a given coin manages the currency itself, except when it is created or cashed. In particular, the user manages all the security features of his/her coin(s). As other micro-payments systems, in PPay coin frauds are possible, but they are unprofitable. More precisely, frauds are detectable and malicious users can be punished as well. Moreover, each fraud concerns only small amounts of currency thus making the benefits not worth the risks. In what follows we provide a short overview of PPay, the interested reader is deferred to [5] for further details.

Table 1. Coins, assignments and re-assignments in PPay

$\gamma = \{X, sn\}_{SK_B}$	(raw coin)
$\lambda_X = \{X, lim_l, lim_u\}_{SK_B}$	(limit certificate)
$\gamma' = \{X, sn\}_{SK_X}$	(user signed raw coin)
$\alpha_{XY} = \{Y, seq_1, \gamma\}_{SK_X}$	(assigned coin)
$\varrho_{XYZ} = \{Z, \alpha_{XY}\}_{SK_Y}$	(re-assignment request)
$\alpha'_{XZ} = \{Z, seq_2, \gamma\}_{SK_X}$	(re-assigned coin)
$\alpha^B_{XZ} = \{Z, seq_2, \gamma\}_{SK_B}$	(broker's reassigned coin)
$\pi_{XYZ} = \{Z, Y, seq_3, \alpha_{XY}\}_{SK_Y}$	(layered coin)

Let X, Y and Z be three users of a p2p system, and let B be the broker. When setting up her own account, a user, say X, purchases digital coins from B. An user can buy a set of **raw coins** γ, signed by B, or a **limit certificate** λ_X that allows her to print her own raw coins γ'. Each raw coin has a serial number sn to detect double spending frauds. The serial number in a coin printed by a user, must belong to the interval defined in the corresponding limit certificate (i.e., $lim_l \leq sn \leq lim_u$).

When X wants to purchase an item or a service from Y, he will send to Y an **assigned coin** α_{XY}, that contains a sequence number of seq_1. The re-assignment of this coin will have a greater sequence number. Now Y is the owner of the coin, and he can decide to cash it or to re-assign it to another user (e.g., Z). In the latter case Y has to send a *reassignment request* ϱ_{XYZ} to X. After receiving ϱ_{XYZ}, X processes it and sends to Y and Z the new assignment α'_{XZ}, containing a sequence number $seq_2 > seq_1$. Of course, after α_{XZ} has been released, α_{XY} is no longer valid.

If X is down when Y wishes to reassign his own coin (or she simply denies to serve Y's request), the **downtime protocol** is used instead: the broker plays the role

of the trusted intermediary, and she generates the newly assigned coin α_{XZ}^{B} in place of X. B sends the reassigned coin to X when this peer comes back on line. This is because X should be responsible for detecting frauds committed when it was off-line. Downtime protocol introduces a drawback due to high percentage of off-line periods in a lifetime of a peer: Broker's load significantly grows up to reassignment requests. Moreover, Broker must continuously check when peers came back on-line, because they must send back the newly assigned coin.

Another reassignment strategy is given by **layered coins**. In this case, Y can re-assign γ itself by sending to Z the assigned coin α_{XY} enveloped in a layer π_{XYZ}. If Z wishes to reassign the coin again, he has to add another layer to π_{XYZ}. Each layer represents a reassignment request and the broker and X can peel off all the layers to obtain all the necessary proofs. Although this protocol is still considered secure, it has the (relatively) negative drawbacks that fraud detection is delayed, and that floating coins grow in size.

3 Fraud Detection Using Aggregate Signatures

Basic re-assignments and layers have both some drawbacks: the former involves mainly the owner of the raw coin, but overloads the broker if she is off-line. The latter lets the coin grow in size, by adding a different signature to each re-assignment; moreover, when a layered coin is finally cashed, the broker has to verify many different signatures. As a consequence, even if this strategy results in better performances w.r.t the basic re-assignment strategy (see [8]), it remains of primary importance to investigate for solution that allows better performances in practice. Ideally, the best re-assignment strategy would be based on a layering scheme where: (a) the coin grows in size as little as possible after each transfer, and (b) the cost of signature verifications does not compromise the broker's efficiency, independently from the number of coins that reach the broker. In this paper we show how to meet both these requirements by using the recently introduced notion of aggregate signatures [7].

3.1 Aggregate Signatures

Aggregate signatures were introduced by Boneh *et al.* [7] to reduce the size of aggregate chains (by aggregating all signatures in the chain) and for reducing message size in secure routing protocols such as SBGP. An implementation of aggregate signatures using bilinear maps was given in [7] and uses the Boneh, Lynn and Shacham signature scheme [9] as underlying building block. Very informally a bilinear map is a function $e : G_1 \times G_2 \rightarrow G_T$ (where G_1, G_2 and G_T are groups) which is linear with respect to both G_1 and G_2. This means that for all integers a, b one has that $e(x^a, y) = e(x, y)^a$ and $e(x, y^b) = e(x, y)^b$. Of course, in order for a bilinear map to be useful in cryptography, some additional properties are required. For the purposes of this it is sufficient to say that "useful" bilinear maps can be constructed from the Weil pairing and the Tate pairing over groups of points of certain elliptic curves. For more details the interested reader is referred to [10].

The rest of this paragraph is devoted to briefly describe the aggregate signature scheme from [7]. For completeness, we give here a more formal definition of the bilinear maps used in cryptography.

BILINEAR MAPS. Let G_1 and G_2 be two cyclic (multiplicative) groups of prime order p. We denote with g_1 a generator of G_1 and with g_2 a generator of G_2. Moreover let ψ be a computable isomorphism from G_1 to G_2, such that $\psi(g_1) = g_2$. Now, let G_T be an additional group such that $|G_T| = |G_1| = |G_2|$. A bilinear map is a map $e : G_1 \times G_2 \to G_T$ with the following properties

1. Bilinear: for all $x \in G_1, y \in G_2$ and $a, b \in \mathbb{Z}$, $e(x^a, y^b) = e(x, y)^{ab}$.
2. Non-degenerate: $e(g_1, g_2) \neq 1$.

Notice that these properties imply that (1) for all $x \in G_1, y_1, y_2 \in G_2$, $e(x, y_1 y_2) = e(x, y_1)e(x, y_2)$ and (2) for any $x, y \in G_1$ $e(x, \psi(y)) = e(y, \psi(x))$.

THE SCHEME. An aggregate signature scheme allows to sign (distinct) messages $M_i \in \{0, 1\}^*$. A signature σ_i is an element in G_2. The groups G_1, G_2, their generators g_1, g_2, the computable isomorphism ψ from G_1 to G_2 and the bilinear map $e : G_1 \times G_2 \to G_T$ (where G_T is the target group), are all system parameters.

The key generation algorithm goes as follows. For each user it picks a random value $x \in \mathbb{Z}_p$, where p is an n-bit prime, and sets $v = g_1^x$ as the user public key. The user secret key is x. A user, holding secret key x, signs a message $M \in \{0, 1\}^*$ as follows. He computes $h = H(M)$ (where H is an hash function modeled as a random oracle mapping elements in $\{0, 1\}^*$ into elements in G_2). The signature is $\sigma = h^x$.

To verify the correctness of a signature σ on a message M, one computes $h = H(M)$ and checks whether $e(g_1, h) = e(v, h)$ holds.

To aggregate ℓ different signatures σ_i (on corresponding *different* messages M_i) one simply computes $\sigma = \prod_{i=1}^{\ell} \sigma_i$. The aggregate signature is $\sigma \in G_2$.

Finally to verify an aggregate signature σ, for the given (different) messages $M_1, \ldots, M_\ell$ and public keys $v_1, \ldots, v_\ell$ one proceeds as follows. First ensure that all the messages are different and reject otherwise. Next, compute $h_i = H(M_i)$ and accept if $e(g_1, \sigma) = \prod_{i=1}^{\ell} e(v_i, h_i)$ holds.

EFFICIENCY ANALYSIS. First notice that a signature is a single point in G_2. As pointed out in [7], on certain elliptic curves these signatures are very short: roughly the half the size of DSA signatures with comparable security. In particular one may set $n = 160$ as security parameter.

To sign one message costs one exponentiation in G_2, which costs $O(n^3)$ bit operations. Thus, signing is roughly 250 times faster than RSA-PSS.

Verification, on the other hand costs two pairing computations. Each pairing computation costs, roughly, 20 modular exponentiations. Thus the cost of verifying a signature is basically $40n^3$. Thus verifying a single signature is, roughly, 150 more expensive than RSA-PSS, with short public exponent.

Aggregation allows to verify ℓ signatures doing $\ell + 1$ pairing computations only. Still, verifying ℓ signatures remains 75 more expensive than RSA-PSS (again, with short public exponent).

4 Modeling Transferable Coins

The goal of modeling PPay is to numerically characterize the life-time of the coins, recalling that they can be printed, transferred and cashed. The computational cost is measured in terms of atomic operations, where an atomic operation is set, by construction, to a RSA digital signature verification.

We describes a framework where a set of peers interact reciprocally sharing items and (re-)assigning coins minted by the broker. We characterize the behavior of the peers in a time interval Δt. We do not make any assumptions about the duration of this time interval. Therefore, in order to simplify our analysis and without altering the results, we can reasonably suppose that all the coins printed during Δt are finally cashed.

Each coin γ is associated to a *re-assignment chain* rc_s^γ during its life-time. Such a chain is made of a sequence of peers, i.e., $rc_s^\gamma = \{p_0^\gamma, p_1^\gamma, \ldots p_s^\gamma\}$, where p_0^γ is the owner of γ, and $\forall i : 0 \leq i < s$, p_i transfers γ to p_{i+1}. Of course, p_s will give the coin back to the broker, to be properly cashed.

We define the *re-assignment limit* m as the maximum length of the re-assignment chain. Hence, $\forall rc_s : s \leq m$. Intuitively, higher m is, more the broker load is decreased, but the detection of double spending frauds is delayed. In the real world is reasonable to set such a boundary.

Let us define $a_0, a_1, \ldots, a_m$ where a_i represents the number of coins reassigned i times and that have been cashed by the broker during the time interval Δt. For example, let us suppose that during Δt, 10 coins are printed. Four of them are never reassigned [2] three are reassigned twice, and other three are reassigned once. If the limit m is set to 3, then we have that $a_0 = 4, a_1 = 3, a_2 = 3$, and $a_3 = 0$. Hence, we observe that C is equal to $\sum_{i=0}^{m} a_i$.

Therefore, we can derive the overall *number of transactions* performed in the market place, namely T, as the sum $\sum_{i=0}^{m}(i+1)a_i$, observing that a coin reassigned i times corresponds to a re-assignment chain of length $i + 1$.

In such a scenario, a meaningful role is played by the distribution of the a_i coefficients. In fact, for a given number of transactions T that take place during Δt, the a_i distribution affects (1) the number of coins printed by the broker, (2) the amount of re-assignments, and, thus, (3) the load of the broker.

Unfortunately, we do not have any idea how users will behave in such a market place, because no one has experimented such technologies in the real world. This has the consequence that neither the analysis parameters can be set in an unique way, nor any empirical measure based on monitored peer-to-peer traffic can be used. Measures performed in the present p2p networks cannot be used in our study, because past analysis (e.g. [2] [11]) were conducted in domains where users download files for free without gaining any profit. Moreover, a fair micro-payment system, should seriously incentive users to reduce the free-riding phenomenon, e.g., as in *FairPeers* [6]. For such reason, we decided to evaluate the entire system making several hypotheses, and comparing reassignment strategies under these different settings.

We focus on two different distributions: *Pareto* and *FullChain*. When $Pareto$ is used, the hypothesis is that an high number of coins will be cashed after few reas-

[2] i.e., These four coins corresponds to a re-assignment chain of length $s = 1$.

Table 2. Cost of atomic actions and modeling parameters

Name	Value	Description
$\|check\|$	1	Verification of one RSA digital signature
$\|gen\|$	$20 \cdot \|check\|$	Generation of one RSA signature
$\|\overline{check}_1\|$	$115 \cdot \|check\|$	Verification of one single aggr. sign.
$\|\overline{check}_\ell\|$	$57,5 \cdot \|check\|$	Verification of ℓ aggr. signatures
$\|\overline{gen}\|$	$5.55 \cdot \|check\|$	Generation of one aggr. sign.
$aggr(\ell)$	$(\ell - 1)/6.4^2$	Aggregation of ℓ signatures ($\ell > 1$)
t	0.8	Off-line peer's rate
f	0.0, 0.05	Frauds rates
lim	10	numbers of coins in a limit certificate

signments, that is likely in the real world. The other distribution, namely $FullChain$, models an optimistic scenario, where each coin is *always* reassigned until it achieves the limit m, i.e., $a_0 = a_1 = a_{m-1} = 0$ and $a_m = C$. Other distributions can be used as well as in [8], but the significance of the results would not change and we did not include other diagrams for the sake of brevity.

Table 2 shows the set of system parameters and the cost of the operations considered in our analysis. The cost of each operation is normalized on the cost of an RSA signature verification ($|check|$). These values are based on the comparison times estimated in [12]. We can observe that the aggregate digital signature scheme looks much more expensive than RSA, except for generation ($|\overline{gen}| < |gen|$).

Moreover, broker's performances are sensibly affected when peers involved in transactions are off-line, and when frauds are detected. Let t be the off-line rate of a peer[3], f the fraud rate. Finally, let lim be the number of coins that a peer's user can print by herself when she owns a limit certificate.

5 Broker's Load Analysis

The broker is engaged when any coin is subjected to the following actions: **printing**, **reassignment**, and **cashing**. These actions affect the broker's performance dependently of the used coin management strategy. As a consequence, we use three functions, ω_P, ω_R, and ω_{Ca}, that respectively define the weights due to printing, reassigning or cashing a coin.

We define the broker's load L_B in terms of these three actions:

$$L_B = C * \omega_P + (T - C) * \omega_R + C * \omega_{Ca} \tag{1}$$

If C and T are, respectively, the number of coins printed and the number of transactions occurred during the given time interval Δt, then $(T-C)$ is the number of reassignments performed during the time interval.

Definitions of ω functions change accordingly to three different coin management aspects: (1) the minting strategy, (2) the reassignment strategy, (3) the adopted digital signature scheme. As introduced in Section 2, there are two different coin *minting*

[3] As in [11], we suppose that a peer is off-line with a 0.8 probability.

strategies: the broker can mint a raw coin for a given peer X or can produce a limit certificate for X. We will refer to these approaches respectively with keywords **Raw** and **Limit**.

We classify *reassignment strategies* with labels **Basic** and **Layer**. In the Basic strategy, each reassignment involves the owner of the coin according to the scheme based on messages ρ and α' (Table 1). If the owner is down, the broker receives the reassignment request from the buyer and sends the reassigned coin α^B to the engaged peers. He has also the charge to contact the owner when he comes back on-line. If Layers are used instead, the coin floats from node to node until the limit m on the number of layers is reached or until a peer decides to cash it. After each hop, the coin grows in size and in number of attached signatures[4].

Finally, our evaluation considers two different *digital signature schemes*: **RSA** and **AS** (Aggregate Signature).

5.1 Computational Analysis

In the following, we compare the cost of all the strategies, identifying each combination with the triplet $[M, R, S]$, where M, R and S represents, respectively, the minting strategy, the reassignment policy, and the signature scheme.

For example, the triplet $[Limit, Basic, AS]$ identifies a system where the broker mints limit certificates, no layers are allowed during reassignment, and messages are signed under the aggregation scheme. Limit certificates represent an attempt to reduce the broker's load by way of allowing a peer to print a coin by himself. In fact, if C is the number of coins that circulates in the system during Δt, and lim is the number of coins that each peer can extract from a limit certificate, then each coin costs $\omega_P = |\overline{gen}| / lim$. Because layered coins are not allowed, the broker is involved during a reassignment with probability t (i.e., when a peer is likely to be off-line). Hence, ω_R is equal to the computational cost of checking a reassignment request ($\left|\overline{check}_1\right|$) and of generating a new assigned coin ($|\overline{gen}|$).

Finally, ω_{Ca} is the sum of the costs of (1) aggregating C signatures ($aggr(C)$), (2) verifying them ($\left|\overline{check}_C\right|$), and (3) detecting of the misbehaving peer(s) in presence of a fraud[5]. The last cost is particularly important, because we have two different scenarios: when we have no frauds ($f = 0$), all the C different signatures are proved valid. But if $f > 0$, the check will simply fail, without identifying the malicious signer(s). In this case, the broker should verify all the signatures one by one, in order to detect the misbehaving peer. For each assigned coin, ω_{Ca} is increased of a value equal to $f \cdot 2 \cdot \left|\overline{check}_1\right|$.

Table 3 shows all the computational costs of the ω functions, for different strategies.

As a final observation, observe that when layered coins are used, then $\omega_R = 0$, because the broker is never involved during reassignments. On the other hand, when a

[4] An hybrid strategy has been analyzed, too. In such a scenario, a peer tries to reassign the coin by way of the owner, but, if the latter is down, the coin is layered instead. We observed that this $Hybrid$ strategy always outperforms the $Layer$ one, even if they differ very slightly. Hence, for the sake of simplicity, we did not show results on this third reassignment strategy.

[5] Observe that the first two components of ω_{Ca} should be divided by the number of aggregated coins (see Table 3) in order to flatten their contribution in formula (1).

Table 3. Values of the weight functions in terms of cryptographic primitives

Strategy	ω_P	ω_R	ω_{Ca}
$[Raw, Basic, RSA]$	$\|gen\|$	$t(3\,\|check\| + \|gen\|)$	$2\,\|check\|$
$[Raw, Layer, RSA]$	$\|gen\|$	0	$(2+s)\,\|check\|$
$[Limit, Basic, RSA]$	$\frac{\|gen\|}{lim}$	$t(3\,\|check\| + \|gen\|)$	$2\,\|check\|$
$[Limit, Layer, RSA]$	$\frac{\|gen\|}{lim}$	0	$(2+s)\,\|check\|$
$[Raw, Basic, AS]$	$\|\overline{gen}\|$	$t(\|\overline{check}_1\| + \|\overline{gen}\|)$	$\frac{aggr(C)+\|\overline{check}_C\|}{C} + 2f\,\|\overline{check}_1\|$
$[Raw, Layer, AS]$	$\|\overline{gen}\|$	0	$\frac{aggr(C)+\|\overline{check}_C\|}{C} + (2+s)f\,\|\overline{check}_1\|$
$[Limit, Basic, AS]$	$\frac{\|\overline{gen}\|}{lim}$	$t(\|\overline{check}_1\| + \|\overline{gen}\|)$	$\frac{aggr(C)+\|\overline{check}_C\|}{C} + 2f\,\|\overline{check}_1\|$
$[Limit, Layer, AS]$	$\frac{\|\overline{gen}\|}{lim}$	0	$\frac{aggr(C)+\|\overline{check}_C\|}{C} + (2+s)f\,\|\overline{check}_1\|$

coin is cashed, it contains many signatures as the number of layers, and ω_{Ca} is higher than in the Basic reassignment strategy. In the estimation of ω_{Ca}, the number of layers in a coin is given by coefficient s, i.e., the length of the reassignment chain of the layered coin. In the next section, the size of layers (i.e., the value of s) is modeled using the *FullChain* and *Pareto* distributions, as previously introduced.

5.2 Results

The scalability of our market place is strictly bound to the load of the broker: less this central unit is overloaded, more the market place is resistant. As a consequence, we want to identify the combination of coin management policies that stresses the broker as less as possible. Hence, for comparing different strategies, we set T to a constant value. A spatial analysis (i.e., bandwidth and storage consumption) is trivial: an RSA signature is sized 128 bytes, against the 20 bytes needed under the AS scheme. As a generalization of [8], the best strategy is $[Limit, Layer, AS]$.

Computational analysis is not so straightforward, and we need an in depth analysis. In order to make our results independent from a given platform, we normalized all the costs at the value of an RSA verification cryptographic operation ($|\overline{check}|$). Hence, the number of $check$ operations is displayed in the y axes in diagrams showed below (Figures 1 and 2). The x axes are instead devoted to the maximum number of reassignment m. In Figure 1 we clearly see that a system adopting aggregate signatures outperforms, even from a computational point of view, an equivalent framework that uses RSA signatures. In fact, if we measure the system performance in terms of number of RSA signature verifications, we observe that when we have no fraudulent peers, and even when the rate of frauds is quite limited ($f < 5\%$), the usage of aggregate signatures appreciably improves the broker's load. This is observed in all the schemes, except for the $[\_, Basic, \_]$ strategies with *FullChain* distribution. However, the Basic reassignment strategy is failing w.r.t. Layer (confirming results in [8]). Observe that, whatever distribution is adopted in the model, the best strategy is definitely $[Limit, Layer, AS]$.

As expected, if the fraud rate grows (e.g., $f = 5\%$ in the diagrams of Figure 2), aggregate signatures deteriorates broker's performance. Anyhow, we can reasonably suppose that the rate of committed frauds will always be very limited, because of the low

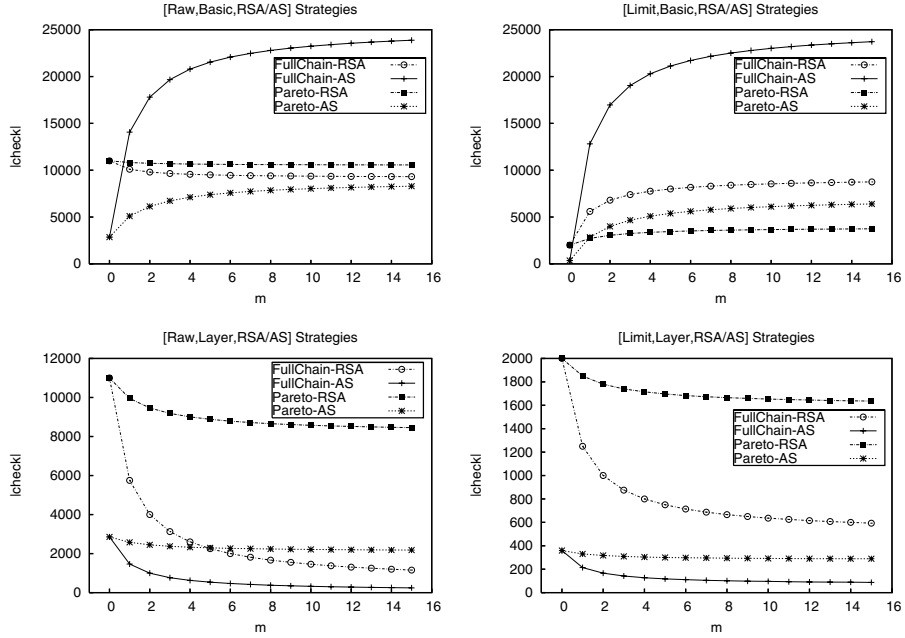

Fig. 1. Computational analysis of different strategies: fraud rate $f = 0\%$

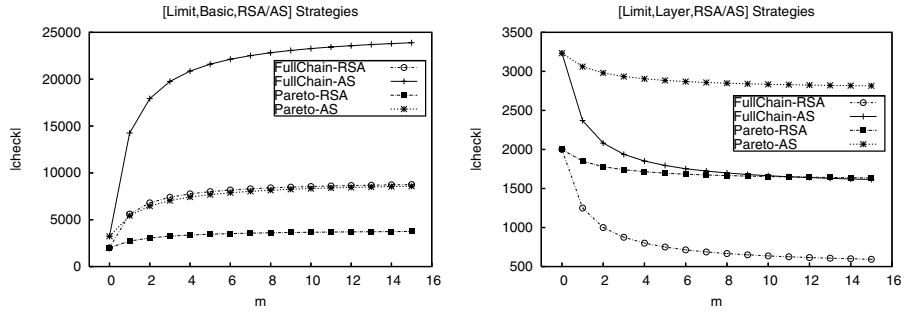

Fig. 2. Computational analysis of different strategies: fraud rate $f = 5\%$

value of the used currency: frauds are detactable and not profitable. In the real world, we can support the adoption of aggregate signatures to verify group of many signatures all at once after a periodical interval of time. Moreover, to enforce results of this study, the reader should observe that we adopted a heavy pessimistic estimation on the performance of cryptographic operations under an AS scheme. An efficient implementation of $\overline{check_\ell}$ would raise up the efficiency threshold based on f.

6 Conclusion

We evaluated the usage of an aggregate signature scheme in the PPay micro-payment system. We proved that aggregate signatures outperform RSA in terms of broker computational and spatial resources consumption. In particular, we showed that the broker's load scales well even if frauds are committed. Future work will focus on a deep analysis, taking advantage of simulators and a prototype implementation.

Acknowledgments

This work has been partially financially supported by the Italian FIRB 2001 project number RBNE01WEJT "Web MiNDS".

References

1. R. Dingledine, M.J.F., D.Molnar: Peer-To-Peer: Harnessing the Power of Disruptive Technologies, Chapter 16. O'Reilly (2001)
2. Adar, E., Huberman, B.A.: Free riding on gnutella. First Monday (2000)
3. Cohen., B.: Incentives build robustness in bittorrent. In: Proc. of the 1st Workshop on the Economics of Peer-to-Peer Systems. (2003)
4. eMule project: (http://www.emule-project.ne)
5. Yang, B., Garcia-Molina, H.: Ppay: micropayments for peer-to-peer systems. In: Proc. of the 10th ACM CCS, ACM Press (2003)
6. Catalano, D., Ruffo, G.: A fair micro-payment scheme for profit sharing in a p2p network. In: Proc. of HOT-P2P 04, IEEE Press (2004)
7. D. Boneh, C. Gentry, B.L., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Eurocrypt '03, LNCS 2656, Springer-Verlag. (2003) 416–432
8. Ruffo, G., Schifanella, R.: Scalability evaluation of a peer-to-peer market place based on micro payments. In: Proc. of HOT-P2P 05, IEEE Press (2005)
9. D. Boneh, B.L., Shacham, H.: Short signatures from the weil pairing. In: Asiacrypt '01, LNCS 2248, Springer-Verlag. (2001) 514–532
10. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Crypto '01, LNCS 2139, Springer-Verlag. (2001) 213–229
11. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: Proc. of SOSP '03, ACM Press (2003)
12. Barreto, P.S.L.M., Lynn, B., Scott, M.: Efficient implementation of pairing-based cryptosystems. J. Cryptology **17** (2004)