Experience Report: Trading Dependability, Performance, and Security through Temporal Decoupling

Lorenz Froihofer, Guenther Starnberger, Karl Goeschka

Experience Report: Trading Dependability, Performance, and Security Through Temporal Decoupling

Lorenz Froihofer, Guenther Starnberger, and Karl M. Goeschka

Vienna University of Technology
Institute of Information Systems, Distributed Systems Group
Argentinierstrasse 8/184-1
1040 Vienna, Austria
{lorenz.froihofer, guenther.starnberger, karl.goeschka}@tuwien.ac.at

Abstract. While it is widely recognized that security can be traded for performance and dependability, this trade-off lacks concrete and quantitative evidence. In this experience report we discuss (i) a concrete approach (temporal decoupling) to control the trade-off between those properties, and (ii) a quantitative and qualitative evaluation of the benefits based on an online auction system. Our results show that trading only a small amount of security does not pay off in terms of performance or dependability. Trading security even more first improves performance and later improves dependability.

Keywords: Temporal decoupling, Dependability, Security, Performance

1 Introduction

While it is widely recognized that security can be traded for performance and dependability [8, 18, 33], this trade-off lacks concrete and quantitative evidence. In this experience report we examine the implementation of a prototype that allows trading these properties in first-price sealed-bid auctions as used for governmental bonds and CO_2 certificates.

Main motivation for examining this auction type are the high dependability and performance requirements as the high monetary amount of traded goods can lead to significant financial losses when auctions need to be canceled or rescheduled. In addition, such auctions typically exhibit a high peak load shortly before the auction's deadline, as a late submission of bids allows bidders to better optimize their bids due to continuously changing financial market conditions. Moreover, cloud computing is not an option due to data ownership issues.

The core idea of our approach is to mitigate performance issues (high peak loads) and dependability problems (fault tolerance of the auctioneer's infrastructure as well as the network infrastructure between client and auctioneer) by shifting them into the security domain and by subsequently solving the new security challenges [26]. In first-price sealed-bid auctions this is possible by decoupling bid submission from bid transmission: Unlike eBay-style auctions, bidders

do not learn about other bids before the auction's deadline. Therefore, we can locally timestamp bids when they are placed on the client and transfer them to the server later. This allows us to increase performance, as we can spread the peak load in the temporal domain, and dependability, as we can reschedule transmission of bids in case of errors. In addition to server-side failures we can also mitigate client-side failures, which cannot be solved with traditional techniques such as redundant server-side infrastructure. This increase of performance and dependability introduces new security challenges as it allows adversaries to attack the locally applied timestamps. Consequently, we introduced a smartcard-based secure timestamping protocol that solves the new security challenges [28].

In addition to our temporal decoupling approach this paper contributes with a quantitative and qualitative evaluation of temporal decoupling as means to trade security for performance and dependability. In this section we first examine the architecture of our prototype, followed by a discussion of the implementation.

1.1 Architecture

This section discusses the architecture of our prototype implementation to make the results presented in the following sections better comprehensible. The server side architecture of the prototype (Figure 1) leverages EJB (Enterprise JavaBean) components and the JBoss Seam framework and is deployed to the JBoss application server. In order to facilitate temporal decoupling we not only leverage a Web browser at the client side, but also additional components:

The first prerequisite to temporal decoupling is a secure smart card running the security-critical parts of the application such as time synchronization and time stamping of bid submissions [25,28]. In order to enable the Web application to talk to the smart card, we introduced the smart card proxy [27], a generic secure approach to enable secure HTTP-based (Hypertext Transfer Protocol) access to smart cards that do not offer an HTTP communication interface.

The second prerequisite is a client side storage facility, provided by Google Gears, to store bids once they are timestamped. This prevents loss of valid bids in case of client crashes so that bids can be sent to the server after client recovery.

1.2 Implementation

The prototype has been developed in two iterations: The first iteration used a Java applet to fulfil the bid submission related tasks. Based on the drawbacks we observed for the Java applet solution, we performed a feasibility study of a Web-only solution (no Java applet required) during the second iteration of prototype development and hence replaced the bid submission tasks with a GWT (Google Web Toolkit) implementation, which has two core responsibilities with respect to our decoupling approach:

- The client side GWT application communicates via the proxy with the smart card in order to perform the time synchronization and time stamping tasks.
- After a bid has been placed, the client side GWT application persistently stores the timestamped bid for later submission to the server. For persistent

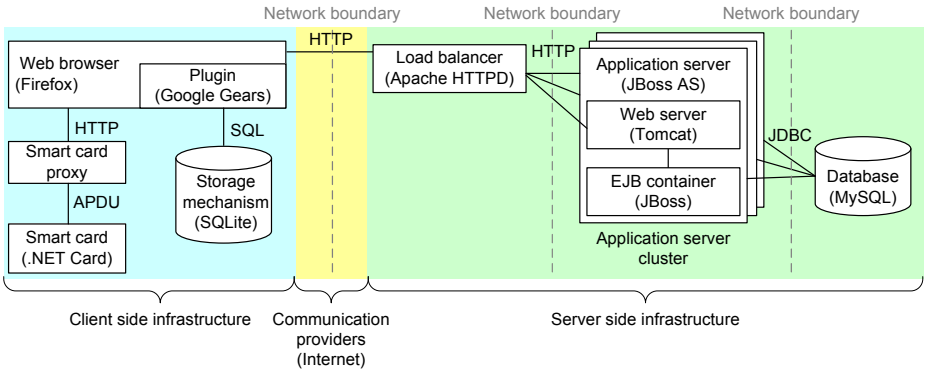


Fig. 1. Prototype architecture

storage, the Google Gears plug-in is required. Finally, the client side GWT application submits bids to the server according to the implemented bid submission strategy [26].

The *smart card proxy* is implemented as an `HttpServlet` according to the Java Servlet specification and is executed in an Apache Tomcat Web application container. It gets requests from the client side GWT application running in the Web browser and performs the necessary translation between the HTTP communication with the GWT application and the APDU (Application Protocol Data Unit) communication with the smart card. The Servlet-based approach has been taken for the proof-of-concept prototype implementation, but could be replaced with any implementation converting from HTTP to the APDU protocol.

2 Evaluation of Temporal Decoupling

The previous section introduced the architecture and implementation of our decoupling approach from a technological perspective. This section goes beyond the technological aspects and evaluates the benefits and drawbacks of the temporal decoupling approach.

Our original idea was to delay all bid submissions for a random period of time, which turned out to be only a sub-optimal solution, because it also delays bid submission in times where the server(s) are not fully loaded. Therefore, we invented and evaluated more sophisticated bid submission strategies [26] in order to better utilize the server infrastructure. The interval-based submission strategy limits clients to the submission of only a single bid within a certain time interval. The group-based submission strategy partitions the set of clients into disjoint groups and only the clients within a specific group might send bids to the server at a specific point in time. In addition, each strategy can be used in two different modes: With bid-queuing all bids submitted at the client are eventually delivered to the server while with bid-overwriting a later bid overwrites any earlier bids still queued at the client.

We discuss the performance improvements through temporal decoupling in this section and compare the two strategies based on the following parameters:

- A bidder issues a bid every 30–40 seconds. This corresponds to real-world data of governmental bond auctions just before the auction deadline.
- 1 500 bidders can be supported by a single server without temporal decoupling. This is the result of the performance measurements of our prototype and specific to our hardware environment.

For analysis and comparison of the temporal decoupling improvement potential we introduce the *decoupling factor* as decoupling period divided by the original peak load duration (parameters illustrated in Figure 2).

The *original peak load duration* is the time between the start of the peak load as it would be observed without our temporal decoupling approach, e.g., determined by exceeding a pre-defined threshold such as average submitted bids per second, and the end of that peak load period, typically close to the deadline for bids. Generally, we assumed five minutes of peak load duration according to our application scenario.

The deadline for bids is the latest point in time where a bid has to be submitted by a bidder while the deadline for messages is the latest point in time where all bids have to be transmitted to the server. The time span between these two deadlines is called *decoupling period*. If we have 5 minutes of original peak load duration and 10 minutes decoupling period, for example, the decoupling factor would be 2.

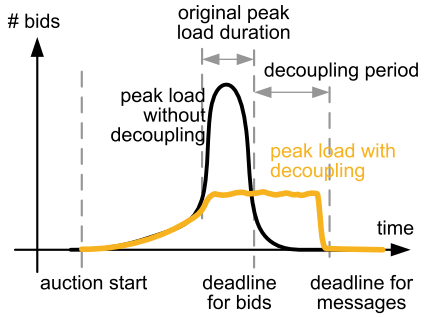


Fig. 2. Temporal decoupling approach

Figure 3 shows the performance improvement possible through temporal decoupling in terms of supportable bidders in relation to the decoupling period (shown before the colon on the X-axis) and the decoupling factor (shown after the colon on the X-axis). In this figure, we assume an original peak load duration of five minutes.

The values of Figure 3 are calculated for the different submission strategies based on the single-server bid submission performance without temporal decoupling. Moreover, we measured the best case scenario in order to verify the upper bound of performance improvement based on temporal decoupling. The results of these investigations are discussed in the following sections.

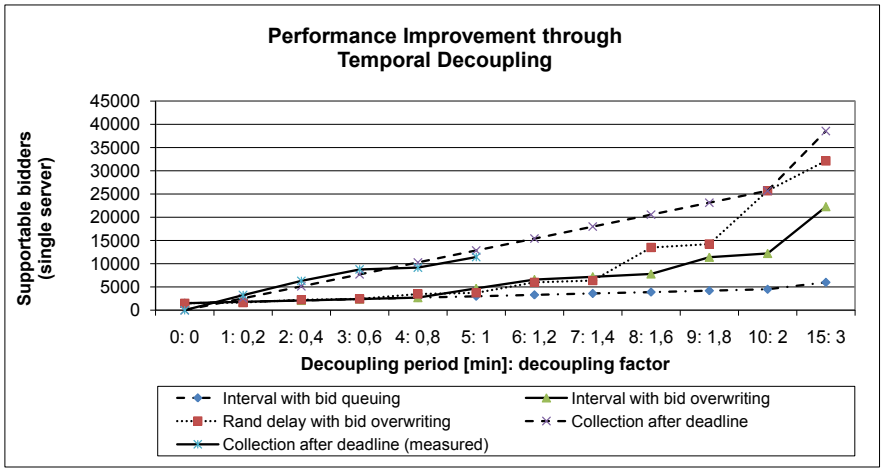
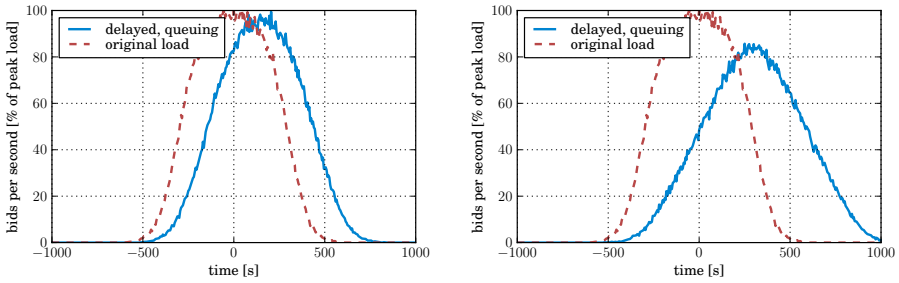


Fig. 3. Performance improvement through temporal decoupling



(a) 5 min peak load with 5 min random delay, decoupling factor = 1

(b) 5 min peak load with 10 min random delay, decoupling factor = 2

Fig. 4. Impact of decoupling parameters

2.1 Worst Case Scenario

The worst case with respect to performance improvements through temporal decoupling is to use a uniform random delay between 0 and the decoupling period together with bid queuing. For a decoupling factor ≤ 1, this only delays the original peak load curve by half of the decoupling period, but does not lead to an efficient load reduction. This behavior is illustrated in Figure 4(a). Figure 4(b) shows that the random delay reduces load, but about 80% of the original peak load for a decoupling factor of two is obviously sub-optimal.

2.2 Best Case Scenario

The best case scenario with respect to performance improvements through temporal decoupling is a completely different submission strategy—the collection of only the latest bid per bidder after the deadline for bids. In this case, no server

side resources are wasted on bids that are later overwritten by subsequent bids and the limit with respect to supportable bidders only depends on the duration of the decoupling period as well the hardware requirements imposed by a specific implementation of this strategy.

Figure 5 illustrates this approach and shows how the load curve for collecting the bids ("peak load with bid collection after deadline") is fully decoupled from the original "peak load without decoupling". Up until the deadline for bids, clients do not submit any bid to the auction servers. After the deadline for bids, the auction servers start to collect the latest bid per client by "asking" the clients to submit their bid. This is performed in a controlled way, so that server overload is avoided.

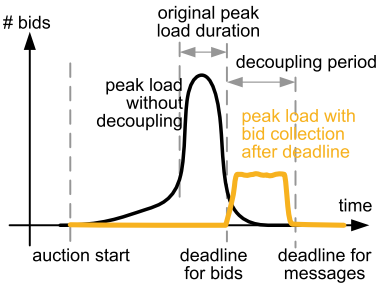


Fig. 5. Collection of all bids after the deadline

A minor optimization to this method would be to start bid collection already shortly before the deadline of bids in a time period where only a single bid can be expected by a single bidder. In our scenario, this would be about 30 seconds before the deadline for bids. However, this would also require to allow for bids to be overwritten in the event that a bidder sends more than one bid within the last 30 seconds before the deadline for bids.

Based on the results of our analysis this approach can already support about 2 600 bidders for a decoupling period of 1 minute and the number of supportable bidders increases linearly with the increase of the decoupling period: within five minutes, about 13 000 bidders can be supported, 26 000 bidders within ten minutes and 39 000 bidders within a decoupling period of 15 minutes on a single server. In practice, hardware requirements and maximum allowable decoupling period limit the total number of supportable bidders.

To verify the calculated performance values, we implemented a version where the clients first issue a request to the server, the server blocks the request until it wants to receive the bid of the specific client, and then notifies the client with the response to the blocked request to submit the bid. Mapped to Web client technologies, this corresponds to an AJAX (Asynchronous JavaScript and XML) push long-polling approach.

Due to the potentially high number of simultaneous open connections to the server, this requires support for asynchronous processing of client requests at the server side as introduced with Java Servlets 3.0 in the Java Enterprise Edition

(JEE) 6, for example. In our case, we used the JBoss-specific asynchronous Servlet API (Application Programming Interface).

The results gained from the prototype measurements correspond quite well to the values expected from calculations and show even better performance. Based on our measurements the numbers of supportable bidders are about 3 200 bidders for a decoupling period of 1 minute, 6 300 bidders for 2 minutes, and 8 700 bidders for 3 minutes. About 8 000 bidders is the reasonable limit for a Pentium 4, 2.8 GHz, and 2 GB of RAM. Measurements with about 10 000 bidders were possible, but for this number nearly all of the RAM was used by the JBoss instance and server-side bid processing was with about 38 processed bid submissions per second already much slower than in the other measurements with a lower number of bidders, which allowed for about 50 bid submissions per second. Based on additional measurements, increasing the RAM increases the number of bidders by about 4 000 bidders per GB of RAM, consumed by the high number of concurrent open client connections.

To increase the number of supportable bidders without increasing the hardware resources would require a different implementation to control bid submission of the clients. Two possible options would be as follows:

- The clients could be partitioned into ordered groups of a specific size and each group gets a specific timeslot for bid submission during the decoupling period. We illustrate this with an example. Let's assume we have a group size of 8 000. From our measurements we know that we need less than 3 minutes to collect the bids of these 8 000 bidders. When initiating the session for a specific client, the client is assigned its group. We start from group number 1 and after we assigned 8 000 clients to group number 1, we start assigning clients to group number 2 and so on. After the deadline for bids, we collect the bids of clients in group 1. Three minutes after the deadline for bids, we collect the bids of clients in group 2. Generally, we collect the bids of clients in group i, $(i-1) \cdot 3$ minutes after the deadline.
- An alternate solution would be to use the distributed feedback channel as detailed in [26]. Unfortunately, firewalls and network address translation (NAT) are a major hindrance for the practical applicability of this solution.

While bid collection only after the deadline for bids delivers the best solution with respect to performance, it also gives an attacker the most time between bid creation and bid transmission. This trade-off between performance and security is further elaborated in Section 2.4.

2.3 Intermediate Approaches

Between the worst case and the best case scenario, we illustrated three other strategies in Figure 3: Interval-based bid queuing, interval-based bid overwriting, and random delay with bid overwriting.

Using the *interval-based bid queuing* strategy, the number of supportable bidders increases linearly with the decoupling factor and can generally be approximated with the following formula: $bidders\_with\_decoupling = bidders\_without\_$

decoupling · (1 + *decoupling_factor*). With this approach, a decoupling factor of two (10 minutes decoupling period for 5 minutes original peak load duration) already allows to support 4 500 bidders and a decoupling factor of 3 allows for 6 000 bidders instead of 1 500 bidders without temporal decoupling for a single server scenario. Obviously, these numbers have to be multiplied by the number of servers used in a server cluster. Therefore, we would be able to support 18 000 bidders with a decoupling factor of 3 on a three nodes server cluster compared to 4 500 bidders without temporal decoupling.

For *interval-based bid overwriting*, Figure 3 shows that this strategy only increases performance for a decoupling factor ≥ 1, compared to the interval-based bid queuing approach. The reason for this is that only in this case bids will be delayed long enough so that bid overwriting will actually take place at the client side.

For the same reason, the interval-based bid overwriting approach does not necessarily have an advantage over the *random delay bid overwriting* approach. In the interval-based approach, the delay of a bid and hence the probability of being overwritten depends on the decoupling factor (relative factor) while in the random delay approach the delay of a bid depends on the decoupling period (absolute factor). This can also be observed from Figure 3, where the random delay bid overwriting approach generally performs better than interval-based bid overwriting. However, this disadvantage could be reduced by requiring the interval-based approach to use time intervals larger than the average time span between two bid submissions of a bidder so that bid overwriting will become generally effective.

2.4 Interpretation

In this section we provide an interpretation of the results presented in the previous sections as well as the insights gained from analysis of the prototype implementation. In particular, we discuss the following trade-offs: Performance vs. security, dependability vs. security, and dependability vs. performance.

Performance vs. Security. As illustrated in Section 2 the "collection of bids after the deadline" variant shows the best performance, but also gives a malicious adversary the most time for attacking the system, when considering the attacks discussed in our trust model [28]. Other approaches allow to specify a maximum delay after which a bid must have been received at the server. For the random delay approach, the maximum time after which a bid must be received at the server is the decoupling period duration. With an interval-based approach, a bid has to be transmitted immediately, if no other bids are queued at the client, or within the first interval for which no bid is already scheduled (which is the immediate next interval in case of bid overwriting). As we have seen, these approaches are only able to achieve sub-optimal performance as they also have to process bids that will be overwritten by subsequent bid submissions.

Concluding, we have a trade-off between optimal performance and optimal security. For optimal performance we would use the bid collection after deadline

strategy, giving a potential attacker the most time to attack the system between bid creation and bid submission. For optimal security, we would send bids immediately, facing the performance problems associated with the original peak load curve. In order to target a solution in the middle, we can control this trade-off by using different submission strategies to delay bid submission together with client side bid overwriting. Which solution approach to take will depend on the security and performance requirements of a specific customer, balanced with the costs of the corresponding hardware requirements.

Dependability vs. Security. If only dependability and security are of concern, but not performance, then bids should be transferred to the server immediately in a healthy system. This prevents attacks on the timestamps applied at the client side. However, if node or link failures occur, then bids should be queued at the client using the bid-overwriting strategy and submitted to the server after the failures are repaired.

If temporal decoupling with bid queuing is applied, the dependability vs. security trade-off is influenced by two further aspects:

- The estimated or proven security of the used smart card as well as the software running on it limits the maximum time span between bid creation and bid transmission due to the requirement to guarantee fair auction conditions to all bidders. If it is possible to (i) crack the smart card or to find out the contained cryptographic keys and (ii) to reverse engineer the software running on the smart card or to find out the used signature mechanism even when no auctions are running, then the temporal decoupling approach must not be applied.
- The accuracy of the smart card's clock determines the maximum offline period where a bidder can still submit bids at the client without time synchronizing the smart card's clock. This is a major issue as it has to be ensured that the smart card's clock cannot be influenced from the outside, as this would considerably decrease the security of the system.

Dependability vs. Performance. The balancing of dependability and performance in our case is a concern for server side clustering, especially in regard to the performance implications of session replication for transparent fail-over in case of server faults. During our evaluation we measured different configurations and compared our prototype's performance with and without session replication as well to an alternate implementation that was implemented with Java ServerFaces (JSF) and Terracotta for session replication. While the performance drawback of session replication for the prototype described in this paper was about 15%, session replication in case of the alternate implementation already reduced performance as soon as a second node was introduced.

With respect to the temporal decoupling approach, however, both dependability and performance benefit from a larger allowable time span between bid creation and bid transmission (more time to collect bids or intermediate repair) and have a disadvantage from a lower time span or the requirement for immediate transmission.

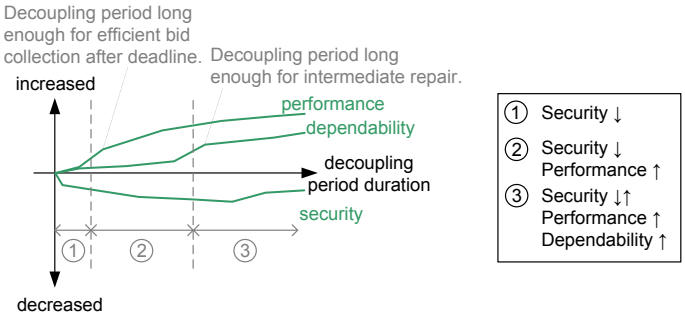


Fig. 6. Influence of decoupling period on different system properties (unquantified)

Balancing Conclusion. Summarizing the previous sections, Figure 6 qualitatively visualizes the different trade-offs, showing that with an increase of the decoupling period, security decreases while dependability and performance increase. The curves in Figure 6 are not quantified on a specific measurement unit as the different properties depend on the specific requirements and threat scenarios of a specific application while we aim at a generic visualization. Based on our investigations and the measurements performed on the prototype, we conclude with the following observations:

1. Security decreases as soon as the temporal decoupling approach is facilitated (Zone 1). How much security decreases depends on the security of the smart card, including the security and accuracy of its clock, as well as an attacker's potential gain from a successful attack along with the reputation loss for the system provider, i.e., the auctioneer in case of our application scenario.
2. A larger decoupling period (Zone 2) improves performance as collection of bids after the deadline becomes efficiently possible. However, there is only a low increase of dependability, as the decoupling period does not yet allow for intermediate repair, e.g., of the auctioneer's Internet connectivity.
3. As soon as the decoupling period is long enough to allow for intermediate repair (Zone 3), dependability is increased significantly. Additionally, security may increase through it's availability attribute if the decoupling period is long enough to submit bids after an intermediate denial of service attack.

Based on our measurements and these observations we recommend to generally facilitate the temporal decoupling approach for scenarios with a reasonably long decoupling period duration, longer than about 4 minutes in our case, as first benefits are only observable after this minimum time span.

3 Related Work

This section presents related work discussing the following interrelations: (i) dependability/security, (ii) dependability/performance, and (iii) security/performance.

3.1 Dependability/Security Interrelation

Research on viewing dependability and security as an integrated concept seems to have started with a focus on intrusion tolerance [10], where research can be traced back to 1985 [11]. One of the first works to establish a common view and terminology on the dependability and security interrelation was published by the IFIP (International Federation for Information Processing) Working Group on Dependable Computing and Fault Tolerance [17]. This publication provides the four dependability attributes availability, reliability, safety, and security, while a later publication in 2004 [3] regards security and dependability to be at the same level and integrates security through the traditional security attributes of confidentiality, integrity, and availability.

Summarizing related work in this area, we observe two main directions for an integrative view on dependability and security:

- Adding dependability means to traditional security mechanisms such as firewalls or cryptographic algorithms, e.g., through redundant/diverse implementations [6, 15, 16, 18].
- Investigation of security issues (intrusions) from a dependability perspective, viewing malicious attacks as faults within a system—with the prominent research area of intrusion tolerance [10, 22, 30, 31].

In contrast to these works, we solve a dependability concern (high availability) through temporal decoupling, thereby shifting it into the security domain (attacks against the client side timestamps) and subsequently mitigate the resulting security challenges.

3.2 Dependability/Performance Interrelation

With respect to the dependability/performance interrelation, degradable systems introduce a grey zone for differentiating between an available and unavailable system state (dependability concern), with different notions of a "slow" system in between (performance concern). Investigation of the interrelation between dependability and performance reveals the following core directions:

- *Performability* as an integrative concept for degradable systems with different performability levels based on which system worth is calculated with a worth function [7, 20, 21].
- *User-perceived availability* takes user sessions and user think time into account when calculating system/service availability. Consequently, only failures during user requests affect availability [13, 14, 19, 23, 32, 34].

While our work allows for a degradable system, i.e., in cases of node or link failures bids might be transferred to the server later, we do not aim at explicit performability levels or user-perceived availability models. The goal in our case is to keep the system parts necessary for secure bid submission available, while tolerating faults in the global infrastructure. Therefore, the system might degrade as bids have to be queued at the client for later delivery, for example, leading to lower user-perceived availability and hence lower performability.

3.3 Security/Performance Interrelation

The interrelation of security with performance [8, 33] is an obvious relationship that has already been researched within different areas:

- Competition for computational resources introduces a trade-off between security, e.g., because of encryption, and performance, e.g., measured in terms of throughput [1, 4, 9, 12].
- Research in the area of packet routing where more complex routing protocols or lower bandwidth result in increased security while showing less performance [2, 29].
- Security/performance trade-off with respect to traffic analysis where higher bandwidth requirements increase traffic flow confidentiality [5, 24].

In contrast to related work, we don't address the security/performance interrelation on a protocol or network level, but trade security for performance with respect to throughput based on architectural trade-offs. For example, we give an attacker more time for an attack in order to increase the number of bidders processable on a given hardware.

Concluding, the interrelations of the different properties have been addressed in several works. However, most of related work explicitly trades only two of the three properties dependability, security, and performance while we contribute with an integrated view on all three for a certain class of application scenarios.

4 Conclusions

In this experience report we showed how the system attributes dependability, security, and performance can be traded in practice by means of temporal decoupling. Based on online auctions as our specific application scenario, we showed how different system requirements demand different trade-offs, addressed through different bid submission strategies in our case. For example, we can achieve higher performance at the price of additional attack possibilities if bids are collected only after the auction deadline.

A considerable amount of research publications addresses trade-offs and interdependencies between dependability, security, and performance already. However, most of these works either focus on the conceptual level or are targeted at a specific trade-off for a certain system component such as security vs. performance in cryptographic algorithms. Although we consider all three properties in a system approach, we focused our investigations of security concerns to the new challenges introduced through temporal decoupling. As our results show a significant potential in explicitly balancing these trade-offs already, we see even more potential in future research on these trade-offs and their – design-time and run-time – balancing from a holistic system perspective.

Acknowledgments. This work has been partially funded by the Austrian Federal Ministry of Transport, Innovation and Technology under the FIT-IT project TRADE (Trustworthy Adaptive Quality Balancing through Temporal Decoupling, contract 816143, `http://www.dedisys.org/trade/`).

References

1. Agi, I., Gong, L.: An empirical study of secure mpeg video transmissions. In: Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96). pp. 137–144. IEEE Computer Society, Washington, DC, USA (1996)
2. Andersen, D.G.: Mayday: Distributed filtering for internet services. In: 4th USENIX Symposium on Internet Technologies and Systems (2003)
3. Avižienis, A., Laprie, J.C., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Sec. Comput. 1(1), 11–33 (2004)
4. Barka, E., Boulmalf, M.: On the impact of security on the performance of wlans. JCM 2(4), 10–17 (2007)
5. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against tor. In: WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society. pp. 11–20. ACM, New York, NY, USA (2007)
6. Chen, Y., He, Z.: Simulating highly dependable applications in a distributed computing environment. In: ANSS '03: Proceedings of the 36th annual symposium on Simulation. p. 101. IEEE Computer Society, Washington, DC, USA (2003)
7. Cho, B., Youn, H., Lee, E.: Performability analysis method from reliability and availability. In: Lee, G., Howard, D., Kang, J.J., Slezak, D., Ahn, T.N., Yang, C.H. (eds.) ICHIT. ACM International Conference Proceeding Series, vol. 321, pp. 401–407. ACM (2009)
8. Cortellessa, V., Trubiani, C., Mostarda, L., Dulay, N.: An architectural framework for analyzing tradeoffs between software security and performance. In: Giese, H. (ed.) ISARCS. LNCS, vol. 6150, pp. 1–18. Springer (2010)
9. Cowan, C., Pu, C., Maier, D., Hintony, H., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q.: Stackguard: automatic adaptive detection and prevention of buffer-overflow attacks. In: SSYM'98: Proceedings of the 7th USENIX Security Symposium. pp. 5–5. USENIX Association, Berkeley, CA, USA (1998)
10. Deswarte, Y., Blain, L., Fabre, J.C.: Intrusion tolerance in distributed computing systems. In: IEEE Symposium on Security and Privacy. pp. 110–121 (1991)
11. Fraga, J., Powell, D.: A fault- and intrusion-tolerant file system. In: Proceedings of the 3rd Intl. Conf. on Computer Security. pp. 203–218 (1985)
12. Haleem, M.A., Mathur, C.N., Chandramouli, R., Subbalakshmi, K.P.: Opportunistic encryption: A trade-off between security and throughput in wireless networks. IEEE Trans. Dependable Secur. Comput. 4(4), 313–324 (2007)
13. Hariri, S., Mutlu, H.: Hierarchical modeling of availability in distributed systems. IEEE Trans. Softw. Eng. 21(1), 50–58 (1995)
14. Kaaniche, M., Kanoun, K., Rabah, M.: A framework for modeling availability of e-business systems. In: Computer Communications and Networks, 2001. Proceedings. Tenth Intl. Conf. on. pp. 40–45 (2001)
15. Komari, I.E., Kharchenko, V., Lysenko, I., Babeshko, E., Romanovsky, A.: Diversity and security of computing systems: Points of interconnection. part 2: Methodology and case study. MASAUM Journal of Open Problems in Science and Engineering 1(2), 33–41 (October 2009)
16. Komari, I.E., Kharchenko, V., Romanovsky, A., Babeshko, E.: Diversity and security of computing systems: Points of interconnection. part 1: Introduction to methodology. MASAUM Journal of Open Problems in Science and Engineering 1(2), 28–32 (October 2009)
17. Laprie (ed.), J. (ed.): Dependability: Basic Concepts and Terminology. Springer (1992)

18. Littlewood, B., Strigini, L.: Redundancy and diversity in security. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS. LNCS, vol. 3193, pp. 423–438. Springer (2004)
19. Mainkar, V.: Availability analysis of transaction processing systems based on user-perceived performance. In: SRDS '97: Proceedings of the 16th Symposium on Reliable Distributed Systems. p. 10. IEEE Computer Society (1997)
20. Meyer, J.F.: On evaluating the performability of degradable computing systems. IEEE Transactions on Computers 29(8), 720–731 (1980)
21. Meyer, J.F.: Performability: a retrospective and some pointers to the future. Performance Evaluation 14(3-4), 139 – 156 (1992), performability Modelling of Computer and Communication Systems
22. Powell, D., Stroud (eds.), R.: Conceptual model and architecture of MAFTIA. Tech. Rep. D21, MAFTIA EU Project (2003)
23. Shao, L., Zhao, J., Xie, T., Zhang, L., Xie, B., Mei, H.: User-perceived service availability: A metric and an estimation approach. In: ICWS. pp. 647–654. IEEE (2009)
24. Snader, R., Borisov, N.: A tune-up for tor: Improving security and performance in the tor network. In: NDSS. The Internet Society (2008)
25. Starnberger, G., Froihofer, L., Goeschka, K.M.: Distributed timestamping with smart cards using efficient overlay routing. In: Fifth Intl. Conf. for Internet Technology and Secured Transactions (ICITST 2010) (Nov 2010)
26. Starnberger, G., Froihofer, L., Goeschka, K.M.: Adaptive run-time performance optimization through scalable client request rate control. In: Proc. 2nd Joint WOSP/SIPEW Intl. Conf. on Performance Engineering (WOSP/SIPEW'11). ACM (March 2011), (to appear)
27. Starnberger, G., Froihofer, L., Goeschka, K.M.: A generic proxy for secure smart card-enabled web applications. In: Web Engineering, 10th Intl. Conf., ICWE 2010, Vienna, Austria, July 5-9, 2010. Proceedings. LNCS, vol. 6189, pp. 370–384. Springer (2010)
28. Starnberger, G., Froihofer, L., Goeschka, K.M.: Using smart cards for tamper-proof timestamps on untrusted clients. In: ARES 2010, Fifth Intl. Conf. on Availability, Reliability and Security, 15-18 February 2010, Kraków, Poland. pp. 96–103. IEEE Computer Society (2010)
29. Timmerman, B.: A security model for dynamic adaptive traffic masking. In: NSPW '97: Proceedings of the 1997 workshop on New security paradigms. pp. 107–116. ACM, New York, NY, USA (1997)
30. Veríssimo, P., Neves, N.F., Cachin, C., Poritz, J.A., Powell, D., Deswarte, Y., Stroud, R.J., Welch, I.: Intrusion-tolerant middleware: the road to automatic security. IEEE Security & Privacy 4(4), 54–62 (2006)
31. Veríssimo, P., Neves, N.F., Correia, M.: Intrusion-tolerant architectures: Concepts and design. In: de Lemos, R., Gacek, C., Romanovsky, A.B. (eds.) Architecting Dependable Systems. LNCS, vol. 2677, pp. 3–36. Springer (2003)
32. Wang, D., Trivedi, K.S.: Modeling user-perceived service availability. In: Malek, M., Nett, E., Suri, N. (eds.) ISAS. LNCS, vol. 3694, pp. 107–122. Springer (2005)
33. Wolter, K., Reinecke, P.: Performance and security tradeoff. In: Aldini, A., Bernardo, M., Pierro, A.D., Wiklicky, H. (eds.) SFM. LNCS, vol. 6154, pp. 135–167. Springer (2010)
34. Xie, W., Sun, H., Cao, Y., Trivedi, K.: Modeling of user perceived webserver availability. In: Communications, 2003. ICC '03. IEEE Intl. Conf. on. vol. 3, pp. 1796–1800 vol.3 (May 2003)